

This Page Is Inserted by IFW Operations  
and is not a part of the Official Record

## **BEST AVAILABLE IMAGES**

Defective images within this document are accurate representations of the original documents submitted by the applicant.

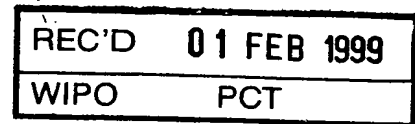
Defects in the images may include (but are not limited to):

- BLACK BORDERS
- TEXT CUT OFF AT TOP, BOTTOM OR SIDES
- FADED TEXT
- ILLEGIBLE TEXT
- SKEWED/SLANTED IMAGES
- COLORED PHOTOS
- BLACK OR VERY BLACK AND WHITE DARK PHOTOS
- GRAY SCALE DOCUMENTS

**IMAGES ARE BEST AVAILABLE COPY.**

**As rescanning documents *will not* correct images,  
please do not report the images to the  
Image Problem Mailbox.**

**THIS PAGE BLANK (USPTO)**



EU

# BREVET D'INVENTION

CERTIFICAT D'UTILITÉ - CERTIFICAT D'ADDITION

## PRIORITY DOCUMENT

SUBMITTED OR TRANSMITTED IN  
COMPLIANCE WITH RULE 17.1(a) OR (b)

COPIE OFFICIELLE

Le Directeur général de l'Institut national de la propriété industrielle certifie que le document ci-annexé est la copie certifiée conforme d'une demande de titre de propriété industrielle déposée à l'Institut.

Fait à Paris, le **05 JAN. 1999**

Pour le Directeur général de l'Institut  
national de la propriété industrielle  
Le Chef du Département des brevets

Martine PLANCHE

INSTITUT  
NATIONAL DE  
LA PROPRIÉTÉ  
INDUSTRIELLE

SIEGE  
26 bis, rue de Saint Petersburg  
75800 PARIS Cédex 08  
Téléphone : 01 53 04 53 04  
Télécopie : 01 42 93 59 30

**THIS PAGE BLANK (USPTO)**

**REQUÊTE EN DÉLIVRANCE**

26 bis, rue de Saint Pétersbourg  
75800 Paris Cedex 08  
Téléphone : 01 53 04 53 04 Télécopie : 01 42 93 59 30

Confirmation d'un dépôt par télécopie ☐

Cet imprimé est à remplir à l'encre noire en lettres capitales

Réservé à l'INPI

DATE DE REMISE DES PIÈCES

30 DEC 1997

N° D'ENREGISTREMENT NATIONAL

97 16699 -

DÉPARTEMENT DE DÉPÔT

75

DATE DE DÉPÔT

30 DEC. 1997

2 DEMANDE Nature du titre de propriété industrielle

☒ brevet d'invention

☐ demande divisionnaire

☐ certificat d'utilité

☐ transformation d'une demande de brevet européen



demande initiale

☐ brevet d'invention

☐ certificat d'utilité n°

date

Établissement du rapport de recherche

☐ différé

☒ immédiat

Le demandeur, personne physique, requiert le paiement échelonné de la redevance

☐ oui

☐ non

Titre de l'invention (200 caractères maximum)

Procédé d'assistance à l'administration d'une application distribuée basée sur un fichier binaire de configuration dans un système informatique.

3 DEMANDEUR (S)

n° SIREN

6 4 2 0 5 8 7 3 9

code APE-NAF

3 0 0 C

Nom et prénoms (souligner le nom patronymique) ou dénomination

BULL S.A.

Forme juridique

Nationalité (s)

Française

Adresse (s) complète (s)

BULL S.A.

68, route de Versailles

78434 LOUVECIENNES

Pays

FRANCE

En cas d'insuffisance de place, poursuivre sur papier libre ☐

4 INVENTEUR (S) - Les inventeurs sont les demandeurs

☐ oui

☒ non

Si la réponse est non, fournir une désignation séparée

5 RÉDUCTION DU TAUX DES REDEVANCES

☐ requise pour la 1ère fois

☐

requise antérieurement au dépôt : joindre copie de la décision d'admission

6 DÉCLARATION DE PRIORITÉ OU REQUÊTE DU BÉNÉFICIAIRE DE LA DATE DE DÉPÔT D'UNE DEMANDE ANTÉRIEURE

pays d'origine

numéro

date de dépôt

nature de la demande

7 DIVISIONS

antérieures à la présente demande

n°

date

n°

date

8 SIGNATURE DU DEMANDEUR OU DU MANDATAIRE

(nom et qualité du signataire - n° d'inscription)

Hervé DENIS (mandataire)

SIGNATURE DU PRÉPOSÉ À LA RÉCEPTION

SIGNATURE APRÈS ENREGISTREMENT DE LA DEMANDE À L'INPI

**DÉSIGNATION DE L'INVENTEUR**

(si le demandeur n'est pas l'inventeur ou l'unique inventeur)

**DIVISION ADMINISTRATIVE DES BREVETS**

26bis, rue de Saint-Petersbourg  
75800 Paris Cédex 08

Tél. : (1) 42 94 52 52 - Télécopie : (1) 42 93 59 30

**FR 3587 HD**

N° D'ENREGISTREMENT NATIONAL

97 16 699

**TITRE DE L'INVENTION :**

Procédé d'assistance à l'administration d'une application distribuée basée sur un  
fichier binaire de configuration dans un système informatique.

**LE (S) SOUSSIGNÉ (S)**

**BULL S.A.**

**DÉSIGNE (NT) EN TANT QU'INVENTEUR (S)** (indiquer nom, prénoms, adresse et souligner le nom patronymique) :

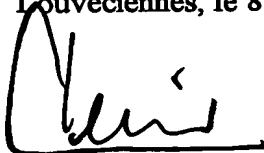
Baillif Christian  
7 bis, avenue du Petit Chambord  
92340 Bourg la Reine  
France

Dia Mama Saidou  
181, avenue Jean Jaurès  
92290 Chatenay Malabry  
France

**NOTA** : A titre exceptionnel, le nom de l'inventeur peut être suivi de celui de la société à laquelle il appartient (société d'appartenance) lorsque celle-ci est différente de la société déposante ou titulaire.

Date et signature (s) du (des) demandeur (s) ou du mandataire

Louveciennes, le 8 janvier 1998



**Hervé DENIS (Mandataire)**

# DOCUMENT COMPORTANT DES MODIFICATIONS

PAGE(S) DE LA DESCRIPTION OU DES REVENDEICATIONS OU PLANCHE(S) DE DESSIN			R.M.*	DATE DE LA CORRESPONDANCE	TAMPON DATEUR DU CORRECTEUR
Modifiée(s)	Supprimée(s)	Ajoutée(s)			
7, 25 à 27	-	28 à 44	R51	2 juin 1998	- 4 JUIN 1998 B E P
15, 17, 20, 23	-	-	-	8 juin 1998	22 JUIN 1998 B E P
3, 10, 19, 21	-	-	-	2 juillet 1998	- 6 JUIL. 1998 B E P

Un changement apporté à la rédaction des revendications d'origine, sauf si celui-ci découle des dispositions de l'article R.612-36 du code de la Propriété Intellectuelle, est signalé par la mention «R.M.» (revendications modifiées).

**Procédé d'assistance à l'administration d'une application distribuée  
basée sur un fichier binaire de configuration dans un système  
informatique**

5           La présente invention concerne un procédé d'assistance à  
l'administration d'une application distribuée basée sur un fichier binaire de  
configuration dans un système informatique. Ce procédé d'assistance à  
l'administration peut notamment être appliqué à un gestionnaire de  
traitement des transactions tel que celui commercialisé sous la marque  
10 "Tuxedo".

L'application "Tuxedo" permet à différents logiciels qui ne se  
connaissent pas mais qui respectent un certain protocole, de travailler  
ensemble.

Généralement, l'application "Tuxedo" est une application distribuée  
15 c'est-à-dire une application qui s'exécute sur plusieurs machines en même  
temps. On appelle "machine", le noeud du réseau au niveau duquel les  
serveurs de l'application "Tuxedo" s'exécutent, et "machine maître" celle  
contrôlant l'application "Tuxedo". La figure 8 illustre le fonctionnement de  
l'application "Tuxedo". Lorsque l'application "Tuxedo" est lancée, le fichier  
20 binaire de configuration (TUXCONFIG) est chargé du disque dans le tableau  
bulletin (Bulletin Board, BB) de la machine maître (Mm). Le tableau bulletin  
(BB) représente un ensemble de structures de données situées dans la  
mémoire partagée et contenant des informations sur les transactions, les  
serveurs, les services et les clients appartenant à l'application "Tuxedo".  
25 Lors du lancement de la machine maître (Mm), le tableau bulletin (BB) est  
chargé dans la mémoire de la machine maître (Mm) à partir du fichier binaire  
de configuration "Tuxedo" (TUXCONFIG). Puis, il est diffusé vers les  
machines esclaves (Me) par le processus maître de l'application appelé  
liaison distinguée du tableau bulletin DBBL (Distinguished Bulletin Board  
30 Liaison). Chaque machine de l'application est sous le contrôle d'un



processus appelé liaison du tableau bulletin BBL (Bulletin Bord Liaison). La liaison distinguée du tableau bulletin DBBL est un processus administratif qui communique avec les processus (BBL), pour coordonner les mises à jour du tableau bulletin (BB). La liaison du tableau bulletin BBL est un processus administratif chargé de tenir à jour une copie du tableau bulletin (BB) sur sa propre machine (Me). Chaque machine (Me) est sous le contrôle d'un processus appelé BBL, défini implicitement par "Tuxedo". Le pont (BRIDGE) (1) est un processus de gestion des communications entre les serveurs de l'application "Tuxedo". Chaque machine est dotée d'un pont défini implicitement par "Tuxedo". Le serveur TMS (Transaction Manager Server) est un processus qui gère un protocole de validation et la reprise pour les transactions exécutées par plusieurs serveurs applicatifs. Le module d'écoute (tlisten, 3) est un processus qui gère les messages destinés à l'application "Tuxedo" sur une machine donnée, avant que le processus pont (BRIDGE) de cette machine n'ait été lancée. Un module d'écoute permet à une machine de recevoir des informations provenant d'autres machines. Un module d'écoute est obligatoire sur chaque machine lorsque l'application est distribuée.

L'application "Tuxedo" est créée par la constitution d'un fichier binaire de configuration qui définit l'architecture de ladite application (figure 7). Lors de la création du fichier de configuration, un administrateur définit les services (Se) fournis par l'application et les assigne à des serveurs (Sr) d'application. L'administrateur définit ensuite des groupes (G) et assigne un ensemble de serveurs (Sr). Enfin, l'administrateur assigne des groupes (G) à une machine (M). Chaque application doit être dotée au minimum d'un groupe (G), d'un service (Se) et d'un serveur (Sr). Une machine (M) peut gérer plusieurs groupes (G).

Après la création d'une application "Tuxedo", celle-ci doit être administrée. L'objet de l'invention est de créer un système d'assistance à l'administration de l'application "Tuxedo". Les principales étapes concernant l'administration d'une application "Tuxedo" consistent en :

- une étape de chargement du fichier binaire de configuration de l'application "Tuxedo" ;

- une étape de lancement des modules d'écoute lorsque l'application "Tuxedo" est une application distribuée ;

5       - une étape de lancement de l'application Tuxedo ;

- une étape de contrôle de l'application. Celle-ci consiste à afficher des informations et à procéder, s'il y a lieu aux corrections requises ;

- une étape d'arrêt de l'application ; et éventuellement

10       - une étape d'arrêt des modules d'écoute lorsque ceux-ci ont été lancés.

L'administration d'une application distribuée peut rapidement devenir très complexe. En effet, avant que cette administration puisse démarrer, l'opérateur doit activer un module d'écoute sur chaque machine esclave sur laquelle il veut agir. Pour cela, l'administrateur doit tout d'abord consulter un  
15       fichier contenant des informations sur l'activation des modules d'écoute. Ce fichier est généralement stocké, à une place dont il faut se souvenir, sur chaque machine. Puis à l'aide de ces informations, l'opérateur doit activer tour à tour le module d'écoute de chaque machine. Ainsi, si l'application concerne dix machines, l'opérateur doit activer le module d'écoute sur les  
20       dix machines, puis à la fin de l'application, désactiver les dix modules d'écoute. Cette opération répétitive est longue et fastidieuse.

Pour effectuer ces tâches, chaque administrateur a sa solution. La solution la plus courante est de stocker sur chaque machine, à une place dont il faut se souvenir des scripts d'activation des modules d'écoute et  
25       d'avoir une copie papier du fichier de configuration. L'administrateur doit veiller à ce que les informations soient à jour à tout moment. A chaque fois que la configuration change, il ne doit pas oublier d'imprimer une copie papier du fichier de configuration et de mettre à jour les scripts sur les machines esclaves.

30       D'autre part, à chaque fois que l'opérateur veut agir sur un élément d'une application, il doit pouvoir identifier rapidement et de façon sûre une

ressource donnée comme par exemple, l'arrêt du serveur "serv1" appartenant au groupe "group1" sur la machine "mach1".

Lorsque le nombre d'application augmente, ces opérations manuelles sont source de nombreuses erreurs.

5        La présente invention a pour but de remédier aux inconvénients de l'art antérieur en proposant un procédé d'assistance à l'administration d'une application distribuée d'un gestionnaire de traitement des transactions, basée sur le fichier binaire de configuration de l'application caractérisé en ce que ledit procédé comporte:

- 10            - une étape de décompilation du fichier de configuration actif de la machine maître,
- une étape de récupération d'informations dans le fichier de configuration décompilé de la machine maître (Mm),
- une étape de vérification de la consistance de ladite application
- 15 mise en oeuvre sur une machine donnée.

Selon une autre particularité, ledit procédé permet de gérer au moins un module d'écoute (3) d'une machine quelconque de l'application à partir d'une autre machine.

20        Selon une autre particularité, les informations concernant ladite application distribuée sont directement prélevées dans le fichier de configuration actif de la machine maître.

25        Selon une autre particularité, l'étape de vérification de consistance de ladite application consiste en une comparaison entre des informations issues du fichier de configuration de la machine maître et des informations issues de ladite application courante mise en oeuvre sur une machine donnée.

30        Selon une autre particularité, ladite gestion des modules d'écoute permet de lancer et d'arrêter au moins un module d'écoute, d'afficher des informations concernant au moins un module d'écoute, de modifier le journal d'au moins un module d'écoute, de vérifier le script d'au moins un module d'écoute et de mettre à jour le script d'au moins un module d'écoute.

Selon une autre particularité, un administrateur se trouvant sur une machine quelconque du réseau peut lancer ou arrêter un module d'écoute mis en oeuvre sur une autre machine du réseau.

5 Selon une autre particularité, ledit procédé permet d'activer plusieurs modules d'écoute en une seule opération.

Selon une autre particularité, une interface graphique facilite la gestion des modules d'écoute.

10 Selon une autre particularité, ladite interface graphique permet de visualiser la structure de ladite application et de sélectionner une valeur voulue dans une liste de valeurs de la configuration courante.

Selon une autre particularité, lorsque le fichier contenant des informations sur ladite application mise en oeuvre sur une machine donnée (tlog) est inexistant, le procédé le génère automatiquement pour pouvoir l'utiliser lors du prochain lancement des modules d'écoute (3).

15 Selon une autre particularité, lesdites informations affichées concernant au moins un module d'écoute comprennent au moins le nom de ladite application, le nom logique de la machine (LMID) sur laquelle ladite application est exécutée, l'identification de l'administrateur (UID) de ladite application, l'adresse utilisé par le module d'écoute (NLSADRR), le chemin  
20 d'accès au réseau de ladite application, le chemin d'accès au fichier journal dudit module d'écoute (LLFPN).

D'autres particularités et avantages de la présente invention apparaîtront plus clairement à la lecture de la description ci-après faite en référence aux dessins annexés dans lesquels :

- 25 - la figure 1 représente une fenêtre de l'interface graphique proposant l'accès aux commandes principales de gestion des modules ;
- la figure 2 représente une fenêtre de l'interface graphique selon la figure 1 permettant d'activer un ou plusieurs modules d'écoute ;
- la figure 3 représente une fenêtre de l'interface graphique selon la  
30 figure 1 permettant l'arrêt d'un ou de plusieurs modules d'écoute ;

- la figure 4 représente une fenêtre de l'interface graphique selon la revendication 1 permettant l'affichage d'informations concernant un module d'écoute d'une application donnée ;

5       - la figure 5 représente une fenêtre de l'interface graphique selon la revendication 1 qui permet de vérifier le script d'un module d'écoute d'une application donnée ;

- la figure 6 représente une fenêtre de l'interface graphique selon la revendication 1 qui permet de mettre à jour le script d'un module d'écoute sur une machine donnée d'une application donnée ;

10       - la figure 7 représente la structure générale d'une application distribuée d'un gestionnaire de traitement des transactions ;

- la figure 8 représente un exemple d'application d'un gestionnaire de traitement des transactions.

15       Suit un exemple non limitatif de spécification de fichier de configuration. Ce fichier de configuration, présenté en annexe 1, concerne l'application "Tuxedo". Il est divisé en sept sections (ressources, machines, groupe, serveur, service, réseau).

20       La section ressource contient des informations générales concernant l'application. Ces informations sont communes à toutes les machines et sont constituées par les paramètres suivants :

- IPCKEY qui représente une clé numérique identifiant le segment de mémoire partagée dans lequel sont stockées les structures d'application. Grâce à cette clé numérique, une application donnée ne peut pas être en conflit avec d'autres applications ;

25       - MASTER qui représente la machine maître ;

- DOMAINID qui représente le domaine de l'application ;

- MAXACCESSERS qui définit le nombre maximum de personnes pouvant accéder à l'application ;

30       - MAXSERVERS qui définit le nombre maximum de serveurs pouvant être rattaché à l'application ;

- MAXSERVICES qui définit le nombre maximum de services pouvant être rattaché à l'application ;

- OPTIONS qui permet de préciser si l'application a lieu sur un réseau local ;

5           - MODEL qui permet de préciser si l'application est distribuée ou si elle ne l'est pas.

La section machines contient des informations sur chaque machine (puce, trifide, zig, orage) du réseau. Ces informations sont constituées par  
10 les paramètres suivants :

- LMID qui définit le nom logique de la machine c'est-à-dire le nom utilisé en interne par l'application, à la place du nom réseau;

de réseau de la machine. Pour simplifier la rédaction du programme, on fait correspondre au nom interne de la machine, un nom logique appelé  
15 "LMID" (Logital Machine ID) ;

- TUXDIR qui spécifie le chemin d'accès au répertoire d'installation du logiciel "Tuxedo" ;

- APPDIR qui spécifie le chemin d'accès aux serveurs applicatifs, c'est-à-dire le chemin menant aux programmes de l'application (par exemple  
20 les programmes concernant l'application "TUXEDO") ;

- TUXCONFIG qui spécifie le chemin d'accès absolu au fichier binaire de configuration TUXCONFIG, celui-ci contenant des informations sur l'application ;

25 - ENVFILE qui spécifie le chemin d'accès au fichier contenant les variables d'environnement pour les serveurs et pour les clients d'une machine donnée;

- ULOGPFX qui spécifie le chemin d'accès au fichier "ULOG" qui contient des informations sur l'historique de l'application.

La section groupe est la section dans laquelle chaque machine est  
30 attribuée à un groupe. Dans l'exemple de l'annexe 1, il existe quatre groupes. Un groupe est un ensemble de serveurs assurant des services

connexes. Dans le cas le plus simple, un groupe n'est constitué que d'un seul serveur. Tous les serveurs d'un groupe doivent s'exécuter sur la même machine. Une application doit comporter au moins un groupe.

La section serveur fournit des renseignements sur chaque serveur.

5 Un serveur est un module fournisseur de services. Dans l'exemple à l'annexe 1, il existe quatre serveurs. Dans le cas le plus simple, un serveur assure un seul service. Une application doit être dotée d'au moins un serveur. La section serveur fournit les renseignements suivants :

- SRVGRP qui définit le groupe auquel le serveur est affilié ;
- 10 - SRVID qui définit le numéro d'identification du serveur ;
- MIN, MAX qui précisent le nombre maximum et minimum d'occurrences de ce serveur;
- RQADDR qui définit le nom de la queue de message utilisée pour l'envoi d'un message ;
- 15 - dans REPLYQ l'administrateur décide de l'existence d'une queue de réponse ;
- CLOPT qui indique les options de démarrage du serveur (services disponibles priorité, ....).

Dans la section service, l'administrateur peut spécifier les services.

20 Un service est un ensemble de fonctions répondant à des requêtes de services émanant d'utilisateurs finaux de l'application. Si l'administrateur désire indiquer des valeurs facultatives différentes des valeurs par défaut, les services doivent obligatoirement être définis.

La section réseau (network) contient pour chaque machine :

- 25 - l'adresse complète utilisée par le processus pont (BRIDGE) appelée "Network Address" ou "NADDR". Les quatre premiers chiffres (0002 dans l'exemple de la figure 4) représentent le protocole de communication utilisé ("tcp" dans l'exemple ci-dessus). Les quatre chiffres suivants représentent le numéro de port utilisé par le processus et les chiffres
- 30 suivants représentent l'adresse réseau de la machine ;

- le chemin d'accès au pont (BRIDGE) de la machine. Le pont est un processus de gestion des communications entre les serveurs de l'application. Il sert à amorcer l'application. Chaque machine est dotée d'un pont.

5           - l'adresse complète du module d'écoute appelée "NLSADDR". Les quatre premiers chiffres représentent le protocole de communication utilisé. Les quatre chiffres suivants représentent le numéro de port utilisé par le module d'écoute qui doit être différent de celui utilisé par le processus pont (BRIDGE). Les chiffres suivants représentent l'adresse réseau de la  
10 machine.

La particularité de l'invention est que les informations concernant l'application sont directement prélevées dans le fichier actif de la machine maître. Un administrateur se trouvant sur une machine quelconque du réseau peut gérer l'exécution de la commande "get\_tuxval" sur la machine  
15 maître pour le compte de l'administrateur comme représenté en page 1 de l'annexe 2.

La sous routine "get\_tuxconfig" du programme utilisé dans la mise en oeuvre du procédé d'assistance à l'administration d'une application distribuée, recherche sur le disque dur de la machine maître le fichier actif  
20 de configuration de l'application. Celui-ci est ensuite décompilé au moyen de la commande "tmunloadcf" (Page 2 de Annexe 2, Lignes 85 à 99).

```

get_tuxconfig() {
    if [ -s tuxconf.tmp.$appname ]
25      then
        cat tuxconf.tmp.$appname
      else
        rm -f tuxconf.tmp.*
        prog="$Env"
30    $TUXDIR/bin/tmunloadcf
    echo "\next $?"
    #print -r "$prog" > prog
        rsh "$MASTER" -l "$ADMIN" "$prog" | tee tuxconf.tmp.$appname
35      fi
get_tlistenlog

```



}

La sous routine "get\_tuxval" de ce programme (Page 2 de l'annexe 2, lignes 112 à 183) prélève les paramètres tels que LMID, APPDIR, TUXCONFIG, TUXDIR, ROOTDIR, ULOGPFX, NLSADDR, UID et BRIDGE  
 5 du fichier binaire de configuration de l'application obtenue à l'aide de la sous routine "get\_tuxconfig".

```

10 get_tuxval() {
    get_tuxconfig | \
    sed -e "s/= /g" -e 's"///g' -e 's/\\V0/g' | awk '
  
```

Les valeurs des paramètres recherchées sont tout d'abord initialisées. Pour cela des matrices associatives appelées  
 15 "tuxconfig\_section" sont créées.

```

BEGIN {
    tuxconfig_section["*RESOURCES"] = 1
    tuxconfig_section["*MACHINES"] = 2
    tuxconfig_section["*GROUPS"] = 3
    20 tuxconfig_section["*SERVERS"] = 4
    tuxconfig_section["*SERVICES"] = 5
    tuxconfig_section["*ROUTING"] = 6
    tuxconfig_section["*NETWORK"] = 7
    }
  
```

25

Un index est associé à chaque matrice. Les paramètres recherchés sont situés dans différentes sections du fichier de configuration. Par exemple pour l'application "Tuxedo", ces différentes sections, au nombre de sept, sont appelées "Ressources", "Machines", "Groupes", "Serveurs",  
 30 "Services" et "Réseau". Pour pouvoir prélever les paramètres dont l'ordinateur a besoin, il doit pouvoir repérer l'endroit où il se trouve dans le fichier de configuration. Dans ce programme, lorsque le nombre de champ (NF) est égal à 1, l'ordinateur se trouve au début d'une section.

```

35 NF == 1 {
    if ( $1 in tuxconfig_section ) {
        section = tuxconfig_section[$1]
    }
  
```

```

    next
  }
}

```

- 5            Si l'ordinateur est dans la section 2 et que le deuxième mot est LMID, l'ordinateur prélève le nom logique de la machine (LMID) sur laquelle l'administrateur se trouve.

```

section == 2 && $2 == "LMID" { # MACHINES section
10  if ( $3 == machine) {
    printf "uname=%s\n", $1
    mach_found=1
  }
  else { # reset mach_found for further machines
15  mach_found = 0
  }
  next
}

```

- 20           Si l'ordinateur est dans la section 2 et que le premier mot est APPDIR, il prélève le chemin d'accès au répertoire sous lequel les serveurs sont amorcés.

```

section == 2 && $1=="APPDIR" && mach_found==1 {
25  printf "appdir=%s\n", $2
    appdir = $2
    next
  }

```

- 30           En procédant de la même manière, l'ordinateur va relever successivement dans la section machine du fichier de configuration le chemin d'accès absolu au fichier binaire de configuration (TUXCONFIG), le chemin d'accès au répertoire d'installation du logiciel Tuxedo (TUXDIR ou ROOTDIR), des informations sur l'historique de l'application (ULOGPFX) et
- 35           dans la section réseau l'adresse du pont de la machine (NLSADDR).

```

section == 2 && $1=="TUXCONFIG" && mach_found == 1 {
    printf "tuxconfig=%s\n", $2

```

```

    next
  }
section == 2 && $1=="TUXDIR" && mach_found==1 {
  printf "tuxdir=%s\n", $2
5   next
  }
section == 2 && $1=="ROOTDIR" && mach_found==1 { # for V4
  printf "tuxdir=%s\n", $2
  next
10  }
section == 2 && $1=="ULOGPFX" && mach_found==1 {
  ulogpfx=1; printf "ulogpfx=%s\n", $2
  next
  }
15 section == 7 && NF == 1 {
  if ( $1 == machine )
    {mach_found = 1}
  else { # reset mach_found for other machines
    mach_found = 0
20  }
  next
  }
section == 7 && $1=="NLSADDR" && mach_found==1 {
  printf "nlsaddr=%s\n", $2
25  next
  }

```

Le programme exécute une boucle sur cette sous routine pour chaque machine jusqu'à ce que l'ordinateur trouve la machine courante. Puis, l'ordinateur se procure dans la section ressources du fichier de configuration l'identification de l'utilisateur de l'application (UID).

```
section == 1 && $1 == "UID" {printf "uid=%s\n", $2 ;next }
```

35 Si aucune valeur n'a été définie pour l'UID dans le fichier de configuration, c'est l'UID de la personne qui a construit l'application qui sera utilisé. Puis, l'ordinateur relève dans la section réseau du fichier de configuration le chemin d'accès au pont (BRIDGE) de la machine.

```
40 section == 7 && $1=="BRIDGE" && mach_found==1 {
```

Le paramètre ULOGPFX représentant l'historique de la machine est une valeur optionnelle. Lorsqu'il est inexistant, l'ordinateur va générer un fichier appelé "ULOG" dans le répertoire APPDIR contenant des informations sur les manipulations opérées sur l'application.

```

5
if ( ulogpfx == 0 ) {
    printf "ulogpfx=%s/ULOG\n", appdir }
    } ' machine=$machine appname=$appname
    lang=`sed -e "s/= /g" -e "s//g" -e "s:/ /" $ConfDir/$appname.tux | awk '
10    $1 == "LANG" {printf "lang=", $2}'`
    }

```

De plus, l'ordinateur a besoin de la langue de travail de l'application représentée par le paramètre LANG, ainsi que de la valeur "tlog". Le  
15 paramètre LANG se trouve dans le fichier de configuration de l'utilisateur.

```

lang=`sed -e "s/= /g" -e "s//g" -e "s:/ /" $ConfDir/$appname.tux | awk '
    $1 == "LANG" {printf "lang=", $2}'`

```

20 La valeur "tlog" fait référence au fichier "tlistenlog . <nom de l'application> . <nom de la machine>" contenant le nom du fichier historique du module d'écoute.

Dans la sous routine get\_tuxval, le programme a rassemblé toutes les variables d'environnement dont il a besoin pour pouvoir lancer le  
25 procédé d'assistance à l'administration d'une application distribuée. Ce procédé permet, en outre d'amorcer et d'arrêter un ou plusieurs modules d'écoute, d'afficher des informations sur un ou plusieurs modules d'écoute, de modifier le journal d'un ou plusieurs modules d'écoute, de vérifier le script d'un ou plusieurs modules d'écoute et enfin de mettre à jour le script d'un ou  
30 plusieurs modules d'écoute (Figure 1).

Le procédé d'assistance à l'administration d'une application "Tuxedo" distribuée est doté d'une interface graphique qui permet l'accès aux commandes du gestionnaire de traitement des transactions. Pour exécuter une tâche, l'administrateur n'est pas tenu d'entrer des commandes,

il lui suffit de cliquer sur des icônes, d'appeler des menus et de spécifier des valeurs via des boîtes de dialogue. Le procédé d'assistance est piloté par menus, structurés sous forme d'arborescence. La sélection d'une option dans le menu principal entraîne l'affichage du menu de niveau inférieur associé. Ce processus est répété jusqu'à l'affichage d'une boîte de dialogue déroulante dans laquelle l'administrateur doit entrer des valeurs de paramètre. Afin de pouvoir gérer les modules d'écoute de l'application "Tuxedo" distribuée, l'administrateur sélectionne à partir du menu principal de "Tuxedo Commands", les fonctions "Tuxedo Commands", "Start/Stop Tuxedo Configuration", "Set up a Tuxedo Application" et "Manage the Listener Processes". Les fonctions sélectionnables "Start Listener Processes", "Stop Listener Processes", "Change/Show Listener Process Parameters", "Show currently running Listener Processes", "Check consistency of Listener Process scripts with TUXCONFIG Level" et "Update Listener Process to TUXCONFIG Level" apparaissent sur la fenêtre de l'interface graphique (Figure 1). Pour lancer des modules d'écoute, l'administrateur doit sélectionner la commande "Start Listener Processes" en positionnant le curseur de sa souris sur le pavé (11) et en appuyant sur le bouton gauche de sa souris. La fenêtre de la figure 2 apparaît après la sélection. Si une application a été préalablement désignée, son nom est affiché sur le pavé (21). Sinon, l'administrateur est informé par la marque clignotante du curseur qu'il doit en donner une. Pour cela, l'administrateur peut soit cliquer sur le bouton "List" (22) pour afficher la liste des applications enregistrées et en sélectionner une, soit entrer explicitement le nom de l'application désirée. Puis l'administrateur est informé par la marque clignotante du curseur dans le pavé (23), à partir de laquelle il doit préciser le nom des machines sur lesquelles un module d'écoute doit être lancé. De la même façon, la liste des machines comprises dans ladite application peut être obtenue en cliquant sur le bouton "List" (22). Pour valider les machines sélectionnées, par exemple par surbrillance, l'administrateur doit cliquer sur le bouton "OK" (24). La commande de démarrage du module d'écoute est

obtenue par la sélection du bouton "Command" (25). Le bouton "Reset" (26) permet de réinitialiser les valeurs des pavés (21) et (23). Le bouton "Cancel" (27) permet d'annuler une valeur introduite sur les pavés (21) et (23). Le bouton "?" (28) offre une aide en ligne à l'administrateur.

- 5 Pour chaque machine désignée dans la liste des machines, l'ordinateur se procure des informations sur l'application dans le fichier de configuration de la machine maître et un fichier historique appelé fichier "tlistenlog . <nom de l'application> . <nom de la machine>" contenant des informations sur l'application agissant actuellement sur cette machine.
- 10 L'ordinateur vérifie d'abord si le module d'écoute n'est pas déjà démarré sur la machine. Si c'est le cas, le message "Listener already running on <nom de la machine>" est imprimé sur l'écran. Sinon, si un fichier local existe, l'ordinateur l'exécute et imprime le message "Listener started on the machine", si la commande réussit. Si la commande échoue, l'ordinateur
- 15 imprime le message "Listener starting failed on <nom de la machine>". Si le fichier local n'existe pas, l'ordinateur génère un fichier "tlistenlog . <nom de l'application> . <nom de la machine>" dans le répertoire APPDIR, l'exécute et rend compte du résultat comme précédemment Ce fichier contient des informations sur l'application courante et sera utilisé dans le prochain
- 20 lancement des modules d'écoute. Ceci correspond aux lignes 652 à 698 de la page 10 et aux lignes 699 à 719 de la page 11 de l'annexe 2.

```
startlistproc)
appname=$1; shift
25     list="$*"
        set_environ
        boucle_status=0
        exit_status=0
        for machine in $list
30     do
        echo "\n----- Machine: $machine ----- \n"
        get_tuxval > "appname.tux"
        get_tllog
        . ./appname.tux
35     prog1="
        TUXDIR=$tuxdir; export TUXDIR
```

```

    ROOTDIR=$tuxdir; export ROOTDIR # V4
    APPDIR=$appdir; export APPDIR
    TUXCONFIG=$tuxconfig; export TUXCONFIG
    PATH=${PATH}:\$TUXDIR/bin:\$APPDIR; export PATH
5    LANG=$lang; export LANG
    LIBPATH=${LIBPATH}:\$tuxdir/lib; export LIBPATH
    COLUMNS=200; export COLUMNS
    ps -eF '%u %p %a' | awk '$3 ~ \"tlisten\" && \$0 ~ \"\$nlsaddr\" {exit 1}'
    if [ \$? = 1 ]
10    then
        echo \"Listener already running on $machine\"
        echo exit 0
        exit 0
    fi
15    if [ -f $appdir/tlisten.$appname.$machine ]
    then
        $appdir/tlisten.$appname.$machine
        ps -eF '%u %p %a' | awk '$3 ~ \"tlisten\" && \$0 ~
    \"\$nlsaddr\" {exit 1}'
20    if [ \$? = 1 ]
    then
        echo \"Listener started on $machine\"
        echo exit 0
    else
25        echo \"Listener starting failed on $machine !!!\"
        echo exit 1
    fi
    else # create the script file & exec it
        echo \"\$tuxdir/bin/tlisten -d $bridge -l $nlsaddr -u $uid -L $tlog\" >
30    $appdir/tlisten.$appname.$machine
        chmod ug+x $appdir/tlisten.$appname.$machine
        $appdir/tlisten.$appname.$machine
        ps -eF '%u %p %a' | awk '$3 ~ \"tlisten\" && \$0 ~ \"\$nlsaddr\" {exit
1}'
35    if [ \$? = 1 ]
    then
        echo \"Listener started on $machine\"
        echo exit 0
    else
40        echo \"Listener starting failed on $machine !!!\"
        echo exit 1
    fi
    fi"
    #echo "$prog1" > prog1
45    if [ -z "$uname" ]
    then
        print "Host $machine not found"
        exit 1
    
```

```

fi
rsh "$uname" -l "$ADMIN" "$prog1" | awk '
NR == 1 {line = $0}
NR > 1 { print line; line = $0 }
END {if(sub("^exit ", "", line)) exit line; print line; exit -1}'
5   boucle_status='expr $boucle_status \|$?' `
done
exit $boucle_status
;;
10

```

Pour arrêter un module d'écoute, l'administrateur sélectionne à partir du menu principal de gestion des modules d'écoute "Manage the Listener Processes", la fonction "Stop Listener Processes" en positionnant son curseur sur la pavé (12) (Figure 1). La fenêtre de la figure 3 apparaît. Elle permet d'indiquer dans un premier pavé (31), le nom de l'application, dans un second pavé (32), le nom de la ou des machines. En cliquant sur le bouton "List" (33), une liste des applications enregistrées ou une liste des machines concernant chaque application peut être obtenue selon la position de la marque de position clignotante (34). Pour chaque machine de l'application, l'ordinateur imprime le nom de la machine pour laquelle le module d'écoute est arrêté. Cette sélection à l'écran grâce à l'interface graphique lance les pas de programmes "stoplistproc" au cours desquels le programme procure à la station sur laquelle la procédure d'arrêt est lancée, des informations par get\_tuxval sur l'application, contenue dans le fichier de configuration de la machine maître (Page 11 de l'Annexe 2, Lignes 720 à 762).

```

stoplistproc)
  appname=$1; shift
  list="$*"
  set_environ
  boucle_status=0
  exit_status=0
  for machine in $list
  do
    echo "\n----- Machine: $machine ----- \n"
    get_tuxval > "appname.tux"
    ../appname.tux
30
35

```



```

prog1="
  COLUMNS=200; export COLUMNS
  ps -eF '%u %p %a' | awk '$3 ~ \"tlisten\" && $0 ~ \"$nlsaddr\" {print $2;
exit 0 }' | read pid
5   if [ -n \"$pid\" ]
      then
          kill -9 $pid > /dev/null
          status=$?
          if [ $status -eq 0 ]
10          then
              echo \"Process $pid killed on $machine\"
              echo exit 0
          else
              echo \"Failed to stop listener on $machine!!!\"
15          echo exit 1
          fi
      else
          echo \"No Listener running on $machine\"
          echo exit 1
20      fi
      if [ -z \"$uname\" ]
          then
              print \"Host $machine not found\"
              exit 1
25          fi
          rsh \"$uname\" -l \"$ADMIN\" \"$prog1\" | awk '
              NR == 1 {line = $0}
              NR > 1 { print line; line = $0 }
              END {if(sub(\"^exit \",\"\", line)) exit line; print line; exit -1}'
30          boucle_status='expr $boucle_status \|$?'
          done
          exit $boucle_status
      ;;

```

35        Si un processus appelé "tlisten" appartenant à l'application courante est en fonctionnement sur cette machine, l'ordinateur l'arrête (kill) et imprime le message "Process <identification du process (PID, Process IDentifier)> killed on <nom de la machine>", sinon il imprime le message "Failed to stop listener on <nom de la machine>".

40        De plus, ce procédé d'assistance à l'administration d'une application permet d'afficher des informations concernant un module d'écoute. Pour cela à partir du menu principal de gestion des modules d'écoute "Manage the

Listener Processes", il suffit à l'administrateur de sélectionner la fonction "Change/Show Listener Processes Parameters" sur le pavé (13) de la fenêtre présentée en Figure 1. La fenêtre de la figure 4 apparaît. L'administrateur doit préciser dans le pavé (41), le nom de l'application et dans le pavé (42), un nom de machine. Suite à cette précision, les autres pavés (43 à 46) de la fenêtre font apparaître les valeurs des paramètres tels que :

- l'identification de l'administrateur (UID),
- l'adresse complète du module d'écoute composée de l'adresse de la machine et du numéro de port qu'il utilise (NLSADDR),
- le chemin d'accès au réseau,
- le chemin d'accès complet au fichier journal du module d'écoute (Listener Logfile Full Path Name, LLFPN),

Toutes ces informations sont extraites du fichier TUXCONFIG de la machine maître. Ces informations ne sont pas modifiables par cette commande, à l'exception du LLFPN. L'annexe 2 présente aux lignes 570 à 579 de la page 9, la partie du programme correspondant à l'exécution de la commande de modification du LLFPN.

```
20  chglisten)
      appname=$1
      machine=$2
      shift 2
      if [ $# -gt 0 ]
25      then
          echo "TLLOG $machine $1" > $ConfDir/tlistenlog.$appname.$machine
      fi
      exit $?
      ;;
30
```

Pour pouvoir visualiser les modules d'écoute actifs de l'application, l'administrateur doit sélectionner la fonction "Show currently running Listener Processes" en cliquant sur le pavé (14) de la fenêtre de la Figure 1. L'ordinateur affiche la liste des machines de l'application sur lesquelles un module d'écoute est actif et l'identification du processus PID (Process

Identifier) appartenant à la configuration du réseau. L'annexe 2 présente aux lignes 764 à 768 de la page 11 et aux lignes 769 à 809 de la page 12, la partie de programme correspondant à l'affichage de la liste des modules d'écoute actifs, qui utilise la fonction get\_tuxval.

```

5
runninglist)
    appname=$1
    boucle_status=0
    set_environ
10    list_lmids=`get_tuxconfig | \
    sed -e "s/= /g" -e 's//g' -e 's/\\V0/' -e "s^*//" | awk '
        BEGIN { network=0 }
        {line = $0}
        NF == 1 { if (network == 1) print $1}
15    $1 == "NETWORK" { network = 1}
    END {if(sub("^exit ", "", line)) exit line; exit -1 }'`
    for machine in $list_lmids
    do
        get_tuxval > "appname.tux"
20    . ./appname.tux
        prog1="
        TUXDIR=$tuxdir; export TUXDIR
        LIBPATH=${LIBPATH}:$tuxdir/lib; export LIBPATH
        ROOTDIR=$tuxdir; export ROOTDIR # V4
25    APPDIR=$appdir; export APPDIR
        TUXCONFIG=$tuxconfig; export TUXCONFIG
        PATH=${PATH}:\$TUXDIR/bin:\$APPDIR; export PATH
        LANG=$lang; export LANG
        COLUMNS=200; export COLUMNS
30    ps -eF %u %p %a | awk '\$3 ~ "tlisten" && \$0 ~ "\$nlsaddr" {print \$2}' |
    read pid
        if [ -n "\$pid" ]
        then
            echo "\Listener running on $machine: pid = \$pid"
35            echo exit 0
        else
            echo "\No Listener running on $machine"
            echo exit 0
        fi
40    if [ -z "$uname" ]
        then
            print "Host $machine not found"
            exit 1
        fi
45    rsh "$uname" -l "$ADMIN" "$prog1" | awk '

```

```

NR == 1 {line = $0}
NR > 1 { print line; line = $0}
END { if (sub("^exit ", "", line)) exit line; print line; exit -1 } '
boucle_status=`expr $boucle_status \|$?`
5  done
  exit $boucle_status
;;

```

L'administrateur peut aussi vérifier le script d'un module d'écoute. En sélectionnant la fonction "Check consistency of Listener Process scripts with Tuxconfig" sur le pavé (15) de la fenêtre représentée en figure 1, la fenêtre de la figure 5 apparaît. L'administrateur doit entrer le nom d'une application sur le pavé (51) et le nom d'une machine donnée sur le pavé (52). Une liste des applications et des machines est à la disposition de l'administrateur grâce au bouton "List" (53). Le programme compare les informations contenues dans le fichier TUXCONFIG de la machine maître et extraites par la fonction "get\_tuxval" avec les informations contenues dans le fichier "tlisten.(nom de l'application).(nom de la machine)" situé dans le répertoire APPDIR de la machine et donne le résultat de cette comparaison. L'annexe 2 présente aux lignes 580 à 631 de la page 9 et aux lignes 632 à 651 de la page 10, la partie du programme correspondant à la vérification d'un script d'un module d'écoute qui permet de signaler les discordances entre les paramètres des fichiers en imprimant par exemple pour le pont "BRIDGE values mismatch".

```

25  chklistscript)
      appname=$1
      machine=$2
      set_environ
      30  get_tuxval > "appname.tux"
      get_tllog
      . ./appname.tux
      prog="
      if [ -f $appdir/tlisten.$appname.$machine ]
      35  then
          cat $appdir/tlisten.$appname.$machine
          echo "\\nexit 0\\n"
      else

```

```

        echo "\"\\nexit 1\"
    fi
    if [ -z "$uname" ]
    then
5       print "Host $machine not found"
        exit 1
    fi
    rm -f tscript.$appname.$machine
    rsh "$uname" -l "$ADMIN" "$prog" | tee tscript.$appname.$machine >
10    /dev/null
    [ $? -ne 0 ] && exit 1
    [ -s tscript.$appname.$machine ] && cat tscript.$appname.$machine |
    awk '
        END { if ( $2 == "1" ) exit -1 } '
15    [ $? -eq -1 ] && exit 1
    [ -s tscript.$appname.$machine ] && cat tscript.$appname.$machine | \
    awk '
        $1 ~ "tlisten" {
            mismatch = 0
            fexec=sprintf("%s/bin/tlisten", tuxdir)
20         if ($1 != fexec) {
                print "tlisten command full pathnames mismatch"
                printf "\tscript:\t%s\n", $1
                printf "\tconfig:\t%s\n", fexec
                mismatch +=1
            }
            for (i=2; i <= NF; i++) {
                if (( $i == "-d" ) && ($(i+1) != bridge)) {
30                 print "BRIDGE values mismatch"
                    printf "\tscript:\t%s\n", $(i+1)
                    printf "\tconfig:\t%s\n", bridge
                    mismatch +=1
                }
                if (( $i == "-l" ) && ($(i+1) != nlsaddr)) {
35                 print "NLSADDR values mismatch"
                    printf "\tscript:\t%s\n", $(i+1)
                    printf "\tconfig:\t%s\n", nlsaddr
                    mismatch +=1
                }
                if (( $i == "-u" ) && ($(i+1) != uid)) {
40                 print "UID values mismatch"
                    printf "\tscript:\t%s\n", $(i+1)
                    printf "\tconfig:\t%s\n", uid
                    mismatch +=1
                }
                if (( $i == "-L" ) && ($(i+1) != tllog)) {
45                 print "LOGFILE values mismatch"
                    printf "\tscript:\t%s\n", $(i+1)

```

```

        printf "\tconfig:\t%s\n", tlog
        mismatch +=1
    }
}
5   END {
    if ( mismatch == 0 )
        printf "Script File is up-to-date for %s\n", machine
    else
        printf "\nScript File is NOT up-to-date for %s\n", machine
10   } ' tlog=$tlog machine=$machine bridge=$bridge \
        nlsaddr=$nlsaddr uid=$uid tuxdir=$tuxdir
    exit $?
    ;;

```

15        Un script d'un module d'écoute peut aussi être mis à jour par la sélection de la fonction "Update Listener Process scripts to TUXCONFIG Level". Un script d'un module d'écoute Tuxedo permet de lancer un module d'écoute. Il suffit d'intégrer un script de ce type pour une machine donnée, dans la séquence de lancement pour que le module d'écoute soit lancé

20        automatiquement en même temps que la machine. Dans la fenêtre représenté figure 6, l'administrateur entre sur le pavé (61) le nom d'une application, et sur le pavé (62) le nom d'une ou de plusieurs machines. Le programme se procure par l'appel de la sous routine "get\_tuxval", toutes les informations dont il a besoin dans le fichier binaire de configuration extraites

25        par la sous routine "get\_tuxconfig" et crée un fichier lui correspondant dans le répertoire APPDIR sous le nom "tlisten.(nom de l'application).(nom de la machine). Les lignes 810 à 831 de l'annexe 2 page 12 présente la partie du programme correspondant à l'exécution de la commande de mise à jour d'un script d'un module d'écoute.

30

```

updtlistscript)
    appname=$1
    machine=$2
    set_environ
35   get_tlog
    get_tuxval > "appname.tux"
    ./appname.tux
    prog="

```

```

    echo \"$tuxdir/bin/tlisten -d $bridge -l $nlsaddr -u $uid -L $tlog\" > $app
dir/tlisten.$appname.$machine
    chmod ug+x $appdir/tlisten.$appname.$machine
    echo exit \"$?\"
5    if [ -z \"$uname\" ]
        then
            print \"Host $machine not found\"
            exit 1
        fi
10    rsh \"$uname\" -l \"$ADMIN\" \"$prog\" | awk '
        NR == 1 {line = $0}
        NR > 1 { print line; line = $0 }
        END {if(sub(\"^exit \", \"\", line)) exit line; print line; exit -1}'
    exit $?
15    ;;

```

D'autres modifications à la portée de l'homme de métier font également partie de l'esprit de l'invention.

## **REVENDECATIONS**

1. Procédé d'assistance à l'administration d'une application distribuée d'un gestionnaire de traitement des transactions, basée sur un  
5 fichier binaire de configuration (TUXCONFIG) caractérisé en ce que ledit procédé comporte:

- une étape de décompilation du fichier de configuration actif de la machine maître,
- une étape de récupération d'informations dans le fichier de  
10 configuration décompilé de la machine maître (Mm),
- une étape de vérification de la consistance de ladite application mise en oeuvre sur une machine donnée.

2. Procédé d'assistance à l'administration d'une application distribuée d'un gestionnaire de traitement des transactions, selon la  
15 revendication 1, caractérisé en ce que ledit procédé permet de gérer au moins un module d'écoute (3) d'une machine quelconque de l'application à partir d'une autre machine.

3. Procédé d'assistance à l'administration d'une application distribuée d'un gestionnaire de traitement des transactions selon la  
20 revendication 1, caractérisé en ce que les informations concernant ladite application distribuée sont directement prélevées dans le fichier de configuration actif de la machine maître.

4. Procédé d'assistance à l'administration d'une application distribuée d'un gestionnaire de traitement des transactions selon la  
25 revendication 1, caractérisé en ce que l'étape de vérification de consistance de ladite application consiste en une comparaison entre des informations issues du fichier de configuration de la machine maître et des informations issues de ladite application courante mise en oeuvre sur une machine donnée.

30 5. Procédé d'assistance à l'administration d'une application distribuée d'un gestionnaire de traitement des transactions s lon la



revendication 2, caractérisé en ce que ladite gestion des modules d'écoute permet de lancer et d'arrêter au moins un module d'écoute, d'afficher des informations concernant au moins un module d'écoute, de modifier le journal d'au moins un module d'écoute, de vérifier le script d'au moins un module d'écoute et de mettre à jour le script d'au moins un module d'écoute.

6. Procédé d'assistance à l'administration d'une application distribuée d'un gestionnaire de traitement des transactions selon la revendication 2, caractérisé en ce qu'un administrateur se trouvant sur une machine quelconque du réseau peut lancer ou arrêter un module d'écoute mis en oeuvre sur une autre machine du réseau.

7. Procédé d'assistance à l'administration d'une application distribuée d'un gestionnaire de traitement des transactions selon la revendication 2, caractérisé en ce que ledit procédé permet d'activer plusieurs modules d'écoute en une seule opération.

8. Procédé d'assistance à l'administration d'une application d'un moniteur transactionnel basée sur un fichier binaire de configuration selon la revendication 2, caractérisé en ce qu'une interface graphique facilite la gestion des modules d'écoute.

9. Procédé d'assistance à l'administration d'une application d'un moniteur transactionnel basée sur un fichier binaire de configuration selon la revendication 9, caractérisé en ce que ladite interface graphique permet de visualiser la structure de ladite application et de sélectionner une valeur voulue dans une liste de valeurs de la configuration courante.

10. Procédé d'assistance à l'administration d'une application distribuée d'un gestionnaire de traitement des transactions selon la revendication 4, caractérisé en ce que lorsque le fichier contenant des informations sur ladite application mise en oeuvre sur une machine donnée (tlog) est inexistant le procédé le génère automatiquement pour pouvoir l'utiliser lors du prochain lancement des modules d'écoute (3).

11. Procédé d'assistance à l'administration d'une application distribuée d'un gestionnaire de traitement des transactions selon la

revendication 6, caractérisé en ce lesdites informations affichées concernant au moins un module d'écoute(3) comprennent au moins le nom de ladite application, le nom logique de la machine (LMID) sur laquelle ladite application est exécutée, l'identification de l'utilisateur (UID) de ladite application, l'adresse utilisée par le module d'écoute (NLSADDR), le chemin d'accès au réseau de ladite application, le chemin d'accès au fichier journal dudit module d'écoute (LLFPN).

Nov 20 1997 16:23:57

ubb.dom1

Page 1

```

1  #
2  #       Tuxedo configuration UBBCONFIG for the model TEST1
3  #
4
5  *RESOURCES
6      IPCKEY                191785
7      MASTER               sitel
8      DOMAINID             dom1
9      MAXACCESSERS         50
10     MAXSERVERS            50
11     MAXSERVICES           100
12     OPTIONS               LAN
13     MODEL                 MP
14
15  *MACHINES
16  puce                    LMID=sitel
17                          TUXDIR="/usr/tuxedo"
18                          APPDIR="/home/dia/tuxedo"
19                          TUXCONFIG="/home/dia/tuxedo/TUXCONFIG"
20                          ENVFILE="/home/dia/tuxedo/envfile_puce"
21                          ULOGPFX="/home/dia/tuxedo/ULOG"
22
23  trifide                 LMID=site2
24                          TUXDIR="/usr/tuxedo"
25                          APPDIR="/home/dia/tmp"
26                          TUXCONFIG="/home/dia/tmp/TUXCONFIG"
27                          ENVFILE="/home/dia/tmp/envfile_trifide"
28                          ULOGPFX="/home/dia/tmp/ULOG"
29
30  zig                     LMID=site3
31                          TUXDIR="/usr/tuxedo"
32                          APPDIR="/home/dia/tuxedo"
33                          TUXCONFIG="/home/dia/tuxedo/TUXCONFIG"
34                          ENVFILE="/home/dia/tuxedo/envfile_zig"
35                          ULOGPFX="/home/dia/tuxedo/ULOG"
36
37  orage                   LMID=site4
38                          TUXDIR="/usr/tuxedo"
39                          APPDIR="/home/dia/tuxedo"
40                          TUXCONFIG="/home/dia/tuxedo/TUXCONFIG"
41                          ENVFILE="/home/dia/tuxedo/envfile_orage"
42                          ULOGPFX="/home/dia/tuxedo/ULOG"
43
44
45  *GROUPS
46
47  DEFAULT:                TMSNAME=TMS      TMSCOUNT=2
48  GROUP1                  LMID=sitel
49                          GRPNO=1
50
51  GROUP2                  LMID=site2
52                          GRPNO=2
53
54  GROUP4                  LMID=site3
55                          GRPNO=3
56
57  GROUP3                  LMID=site4
58                          GRPNO=4
59
60  *SERVERS
61  #
62  DEFAULT: RESTART=Y MAXGEN=5 REPLYQ=Y CLOPT="--A"
63
64  SRV1                    SRVGRP=GROUP1
65                          SRVID=100
66                          MIN=2    MAX=2
67                          RQADDR=QSRV1_1
68                          REPLYQ=Y
69                          CLOPT="--s SVC1_1 -s SVC1_2 -- "
70
71  SRV2                    SRVGRP=GROUP2

```

## ANNEXE 1

Feuille avant rectification

Nov 20 1997 16:23:57

ubb.dom1

Page 2

```
72          SRVID=200
73          MIN=2    MAX=2
74          RQADDR=QSRV2_2
75          REPLYQ=Y
76          CLOPT="-s SVC2_1 -s SVC2_2 -- "
77  SRV4
78          SRVGRP=GROUP4
79          SRVID=300
80          MIN=2    MAX=2
81          RQADDR=QSRV4_3
82          REPLYQ=Y
83          CLOPT="-s SVC4_1 -s SVC4_2 -- "
84  SRV3
85          SRVGRP=GROUP3
86          SRVID=400
87          MIN=2    MAX=2
88          RQADDR=QSRV3_4
89          REPLYQ=Y
90          CLOPT="-s SVC3_1 -- "
91
92
93  *SERVICES
94  DEFAULT:      LOAD=50
95  SVC1_1
96  SVC1_2
97  SVC2_1
98  SVC2_2
99  SVC4_1
100 SVC4_2
101 SVC3_1
102
103
104
105 *NETWORK
106 site1
107 #      port number=60951 (ee17 hexa)
108 #      local address=81b683e0
109 #      NADDR="\x0002ee1781b683e00000000000000000"
110 #      BRIDGE="/dev/xti/tcp"
111 #      port number=60952 (ee18 hexa)
112 #      NLSADDR="\x0002ee1881b683e00000000000000000"
113 #
114 site2
115 #      port number=60951 (ee17 hexa)
116 #      local address=81b68387
117 #      NADDR="\x0002ee1781b683870000000000000000"
118 #      BRIDGE="/dev/xti/tcp"
119 #      port number=60952 (ee18 hexa)
120 #      NLSADDR="\x0002ee1881b683870000000000000000"
121 #
122 site3
123 #      port number=60951 (ee17 hexa)
124 #      local address=81b683e1
125 #      NADDR="\x0002ee1781b683e10000000000000000"
126 #      BRIDGE="/dev/xti/tcp"
127 #      port number=60952 (ee18 hexa)
128 #      NLSADDR="\x0002ee1881b683e10000000000000000"
129 #
130 site4
131 #      port number=60951 (ee17 hexa)
132 #      local address=81b6838b
133 #      NADDR="\x0002ee1781b6838b0000000000000000"
134 #      BRIDGE="/dev/xti/tcp"
135 #      port number=60952 (ee18 hexa)
136 #      NLSADDR="\x0002ee1881b6838b0000000000000000"
137 #
138
```

```

1
2 # @BULL_COPYRIGHT@
3 #
4 # HISTORY
5 # $Log: smtuxadmin.ksh,v $
6 # Revision 1.7 1996/02/12 11:40:49 odeadm
7 #     bci V1Set2C 23.01.96
8 #     [1996/01/23 14:31:07 dia]
9 #
10 # Revision 1.6 1995/12/20 14:26:59 odeadm
11 #     V1 Set2: Still troubles with smtuxadmin.ksh
12 #     [1995/12/11 11:56:55 odeadm]
13 #
14 #     07.12.95 V1Set2 first batch of corrections
15 #     [1995/12/07 17:22:57 odeadm]
16 #
17 #     *** empty log message ***
18 #     [1995/11/30 13:48:30 dia]
19 #
20 #     *** empty log message ***
21 #     [1995/11/30 13:48:30 dia]
22 #
23 # Revision 1.5 1995/10/13 11:52:51 odeadm
24 #     Servers TMS/Partitioned mach.
25 #     [1995/10/09 12:05:57 dia]
26 #
27 # Revision 1.4 1995/09/15 15:15:06 odeadm
28 #     Corrections MRs BUILD 3
29 #     [1995/09/07 15:45:27 dia]
30 #
31 # Revision 1.3 1995/08/24 13:38:03 odeadm
32 #     Build3
33 #     [1995/08/23 09:04:31 odeadm]
34 #
35 # Revision 1.2 1995/07/19 15:18:13 odeadm
36 #     Madison build M0.2
37 #     [1995/07/10 10:01:58 odeadm]
38 #
39 # $EndLog$
40 #! /bin/ksh
41 ConfDir=$WRAPPING_CONFIGURATION
42 Context=smtuxedo.ctx
43 Scanconf=$MADISON_VAR/surveyor/scanconf.tux
44 V5_to_V4='ROOTDIR=$TUXDIR; export ROOTDIR'
45 Set1_to_Set2='[ -z "$ADMIN" ] && export ADMIN="madison"'
46 cmd=$1; shift

set_environ() {
49     MASTER=""; APPDIR=""; ADMIN=""
50     filename=$ConfDir/$appname.tux
51     Env=`tuxgetenv -k -v APP_PW $filename << !
52     tuxgetenvp
53     !`
54     eval "$Env"; unset APP_PW
55     eval "$Set1_to_Set2"
56     if [ -n "$MASTER" -a -n "$APPDIR" ]
57     then
58         Env="$Env
59 $PW
60 $Set1_to_Set2
61 $V5_to_V4"
62 LD_LIBRARY_PATH=$LIBPATH; export LD_LIBRARY_PATH;
63 cd $APPDIR
64 PATH=${PATH}::$APPDIR:$TUXDIR/bin; export PATH'
65         return 0
66     fi
67     exit 1
68 }

70 remote_cmd() {
71     prog="$Env

```

## ANNEXE 2

```

72 $cmd""
73 status=?
74 sleep 1
75 echo "\nexit $status"
76 '
77 #print -r "$prog" > prog
78   rsh "$MASTER" -l "$ADMIN" "$prog" | awk '
79       NR == 1 {line = $0}
80       NR > 1 { print line; line = $0}
81       END {if(sub("^exit ","", line)) exit line; exit -1 }'
82 )
83
84
85 get_tuxconfig() {
86     if [ -s tuxconf.tmp.$appname ]
87     then
88         cat tuxconf.tmp.$appname
89     else
90         rm -f tuxconf.tmp.*
91         prog="$Env"
92 $TUXDIR/bin/tmunloadcf
93 echo "\nexit $?"
94 '
95 #print -r "$prog" > prog
96   rsh "$MASTER" -l "$ADMIN" "$prog" | tee tuxconf.tmp.$appname
97   fi
98 get_tlistenlog
99 }
100
101 get_tlistenlog() {
102     tllogfname=$ConfDir/tlistenlog.$appname.$machine
103 if [ -s $tllogfname ]
104 then
105     cat $tllogfname
106 else # default value
107     echo "TLLOG $machine $MADISON_TMP/tlisten.$appname.$machine.log" | tee $tllogfname
108 fi
109 echo "\nexit $?"
110 }
111
112 get_tuxval() {
113     get_tuxconfig | \
114     sed -e "s/=//g" -e 's/"//g' -e 's/\\\\/0/g' | awk '
115 BEGIN {
116     tuxconfig_section["*RESOURCES"] = 1
117     tuxconfig_section["*MACHINES"] = 2
118     tuxconfig_section["*GROUPS"] = 3
119     tuxconfig_section["*SERVERS"] = 4
120     tuxconfig_section["*SERVICES"] = 5
121     tuxconfig_section["*ROUTING"] = 6
122     tuxconfig_section["*NETWORK"] = 7
123 }
124 NF == 1 {
125     if ( $1 in tuxconfig_section ) {
126         section = tuxconfig_section[$1]
127         next
128     }
129 }
130 section == 2 && $2 == "LMID" { # MACHINES section
131 if ( $3 == machine) {
132     printf "uname=%s\n", $1
133     mach_found=1
134 }
135 else { # reset mach_found for further machines
136     mach_found = 0
137 }
138 next
139 }
140 section == 2 && $1=="APPDIR" && mach_found==1 {
141     printf "appdir=%s\n", $2
142     appdir = $2

```

```

143     next
144     }
145     section == 2 && $1=="TUXCONFIG" && mach_found == 1 {
146         printf "tuxconfig=%s\n", $2
147     next
148     }
149     section == 2 && $1=="TUXDIR" && mach_found==1 {
150         printf "tuxdir=%s\n", $2
151     next
152     }
153     section == 2 && $1=="ROOTDIR" && mach_found==1 { # for V4
154         printf "tuxdir=%s\n", $2
155     next
156     }
157     section == 2 && $1=="ULOGPFX" && mach_found==1 {
158         ulogpfx=1; printf "ulogpfx=%s\n", $2
159     next
160     }
161     section == 7 && NF == 1 {
162         if ( $1 == machine )
163             {mach_found = 1}
164         else { # reset mach_found for other machines
165             mach_found = 0
166         }
167     next
168     }
169     section == 7 && $1=="NLSADDR" && mach_found==1 {
170         printf "nlsaddr=%s\n", $2
171     next
172     }
173     section == 1 && $1 == "UID" {printf "uid=%s\n", $2 ;next }
174     section == 7 && $1=="BRIDGE" && mach_found==1 {
175         printf "bridge=%s\n", $2 }
176     END { # not defined ulogpfx
177         if ( ulogpfx == 0 ) {
178             printf "ulogpfx=%s/ULOG\n", appdir )
179             ' machine=$machine appname=$appname
180             lang=`sed -e "s/=//g" -e "s/'//g" -e "s/;/ /" $ConfDir/$appname.tux | awk '
181             $1 == "LANG" {printf "lang=", $2}' `
182         }
183     }
184     get_tllog() {
185     tllogfname="$ConfDir/tlistenlog.$appname.$machine"
186     if [ -f $tllogfname ]
187     then
188         tllog=`cat $tllogfname|awk '$1 == "TLLOG" && $2 == machine { print $3 }' machine=$m
189         achine`
190     else
191         tllog="$MADISON_TMP/tlistenlog.$appname.$machine"
192         echo "TLLOG $machine $tllog" > $tllogfname
193     fi
194     }
195
196     case $cmd in
197         appli)
198             ls -l $ConfDir 2> /dev/null | awk '
199                 sub(".tux$", "", $NF) {print $NF}'
200             ;;
201         isexist)
202             if [ -f $ConfDir/$1.tux ]
203             then
204                 echo "Yes"
205             else
206                 echo "No"
207             fi
208             ;;
209         setparam)
210             [ ! -d $ConfDir ] && mkdir -p $ConfDir
211             if [ -n "$2" ]
212             then

```

# ANNEXE 2

```

213         filename=$ConfDir/$2.tux
214         while [ $# -gt 0 ]
215         do
216             echo "$1=\""$2\""; export $1"
217             shift 2
218         done > $filename
219     fi
220 ;;
221 discover)
222     [ -z "$1" ] && exit 1
223     filename=$ConfDir/$1.tux; shift
224     if [ -f $filename ]
225     then
226         #
227         awk '
228             BEGIN { field = "#promptW:promptP:promptPO:promptS:promptA:pr
229             omptM:promptC:promptR:promptF"; value=":::::::::" }
230             /\=/ {
231                 for (i=1; i<= NF; i++) {
232                     if (sub("=", "", $i)) {
233                         separator = ":"
234                         field = field separator $i
235                         value = value separator $(i+1)
236                     }
237                 }
238             }
239             END {
240                 print field; print value
241             }' FS=" "
242         else
243             print '#\n'
244         fi
245     ;;
246 delappname)
247     if [ -n "$2" ]
248     then
249         filename=$ConfDir/$2.tux
250         if [ -f $filename ] && grep -q "$1=['\""]*$2" $filename
251         then
252             rm -f $filename ${filename}p
253         else
254             echo 'The file does not exist'
255             echo '      or'
256             echo 'The file is not an environment file'
257             exit 1
258         fi
259     fi
260 select)
261     if [ -n "$2" ]
262     then
263         echo "$1='$2'; export $1" > "$Context"
264     fi
265 ;;
266 deselect)
267     rm -f "$Context"
268 ;;
269 selected)
270     APPNAME=""
271     [ -f $Context ] && . ./Context
272     echo "$1$APPNAME"
273 ;;
274 isselected)
275     rm -f tuxconf.tmp.*
276     [ -f $Context ] && fgrep -q "APPNAME=" $Context && shift
277     echo $1
278 ;;
279 loadcf)
280     appname=$1

```



```

281     boucle_status=0
282     cmd="\$TUXDIR/bin/tmloadcf -y \$2 \$3"
283     set_environ
284     echo "---- Loading Configuration Binary File ----"
285     remote_cmd
286     status=$?
287     if [ \$status -ne 0 ]
288     then
289         exit \$status
290     else
291 # maj fichier $Scanconf.tux machines
292     prog="$Env"
293     $TUXDIR/bin/tmunloadcf
294     echo "\nexit $?"
295
296     #print -r "$prog" > prog
297     rsh "$MASTER" -l "$ADMIN" "$prog" > tuxconf.tmp.$appname
298     list_lmids='cat tuxconf.tmp.$appname | sed -e "s/= / /g" -e "s/"/ /g" -e "s/\*//
" | awk '
{line = $0}
$2 == "LMID" && machine == 1 {lmids = lmids $3 " "; next}
$1 == "GROUPS" && $2 == "" { machine=0; next}
$1 == "MACHINES" && $2 == "" { machine = 1; next}
END {if(sub("^exit ", "", line)) {
    print lmids
    exit line}
    exit -1 }'
for machine in $list_lmids
do
    echo "---- Updating $Scanconf on $machine ----\n"
    get_tuxval > "appname.tux"
    . ./appname.tux
    log_prefix='echo $ulogpfx | sed -e 's./ .g' | awk '
{print $NF}'
    log_dir='echo $ulogpfx | sed -e 's./ .g' | awk '
{for (i=1; i< NF; i++) {
    tempo = tempo "/" $i }}
END { print tempo}'
#Build the 3 lines of $Scanconf for the application
    prog="
[ -x $MADISON_BIN/security/updscantux ] &&
$MADISON_BIN/security/updscantux $appname $log_dir $log_prefix
echo "\n\nexit \${?}"
    rsh "$uname" -l madison "$prog" | awk '
NR == 1 {line = $0}
NR > 1 { print line; line = $0}
END {if(sub("^exit ", "", line)) exit line; exit -1 }'
    boucle_status='expr $boucle_status + $?'
done
fi
exit $boucle_status
;;
apppwd)
    filename=$ConfDir/$1.tuxp
    echo "Enter Application Password: \c"
    OLDCONFIG='stty -g'
    stty -echo
    read APP_PW
    echo "\nRe-enter Application Password: \c"
    read APP_PW_1
    stty $OLDCONFIG
    if [ "$APP_PW" != "$APP_PW_1" ]
    then
        echo "\n\nPassword mismatch!"
        echo "Enter any character to exit and retry"
        read
    else
        PWencode "APP_PW=\"\$APP_PW\"; export APP_PW" > $filename
        APP_PW='echo $APP_PW | sed -e "s/\'/\'"/g'
        PWencode "APP_PW='\$APP_PW'; export APP_PW" > $filename
    fi
tuxgetenv -s > $filename << !

```

```

351 tuxgetenvp
352 $APP_PW
353 !
354     fi
355     ;;
356     chksyntx)
357         appname=$1
358         cmd="\$TUXDIR/bin/tmloadcf -n $2"
359         set_environ
360         remote_cmd
361         exit $?
362     ;;
363     dispIpc)
364         appname=$1
365         cmd="\$TUXDIR/bin/tmloadcf -c $2"
366         set_environ
367         remote_cmd
368         exit $?
369     ;;
370     machine_network)
371         appname=$1
372         set_environ
373         get_tuxconfig | \
374         sed -e "s=/ /g" -e 's/"//g' -e 's/\\// ' -e "s/\*//" | awk '
375             BEGIN { network=0 }
376             {line = $0}
377             NF == 1 { if (network == 1) print $1}
378             $1 == "NETWORK" { network = 1}
379             END {if(sub("^exit ","", line)) exit line; exit -1 }'
380         exit $?
381     ;;
382     machine_machines)
383         appname=$1
384         set_environ
385         get_tuxconfig | \
386         sed -e "s=/ /g" -e 's/"//g' -e 's/\\// ' -e "s/\*//" | awk '
387             BEGIN { machine=0 }
388             {line = $0}
389             $2 == "LMID" { if(machine == 1) print $3}
390             $1 == "GROUPS" { if( $2 == "" ) machine=0}
391             $1 == "MACHINES" { if( $2 == "" ) machine = 1}
392             END {if(sub("^exit ","", line)) exit line; exit -1 }'
393         exit $?
394     ;;
395     group)
396         appname=$1
397         set_environ
398         get_tuxconfig | \
399         sed -e "s=/ /g" -e 's/"//g' -e 's/\\// ' -e "s/\*//" | awk '
400             BEGIN { group=0 }
401             {line = $0}
402             $1 == "SERVERS" { group=0 }
403             $1 == "GROUPS" { if($2 == "") group=1}
404             $2 == "LMID" && $4 == "GRPNO" { if(group) print $1}
405             END {if(sub("^exit ","", line)) exit line; exit -1 }'
406         exit $?
407     ;;
408     svrname)
409         appname=$1
410         set_environ
411         get_tuxconfig | \
412         sed -e "s=/ /g" -e 's/"//g' -e 's/\\// ' -e "s/\*//" | awk '
413             BEGIN { group=server=nb_of_distinct_svr_name=0 }
414             {line = $0}
415             $1 == "TMSNAME" { if ( group == 1 ) {
416                 trouve = 0
417                 if (nb_of_distinct_svr_name == 0) {
418                     nb_of_distinct_svr_name = 1
419                     svr_names[nb_of_distinct_svr_name] = $2
420                     print $2
421                 }
422             }

```

```

422     } else {
423         for (j=1; j<= nb_of_distinct_svr_name; j++) {
424             if ( $2 == svr_names[j] ) {
425                 trouve=1
426             }
427         }
428         if (trouve == 0) {
429             nb_of_distinct_svr_name += 1
430             svr_names[nb_of_distinct_svr_name] = $2
431             print $2
432         }
433     }
434 }
435
436 $1 == "SERVERS" { if ($2 == "") {
437     server=1
438     group=0 }
439 }
440 $1 == "SERVICES" { if ($2== "") server=0}
441 $1 == "GROUPS"    { if ($2 == "") group=1}
442 $2 == "SRVGRP" {
443     if((server == 1) && ($4 == "SRVID")) {
444         trouve = 0
445         if (nb_of_distinct_svr_name == 0) {
446             nb_of_distinct_svr_name = 1
447             svr_names[nb_of_distinct_svr_name] = $1
448             print $1
449         } else {
450             for(j=1; j<= nb_of_distinct_svr_name; j++) {
451                 if ( $1 == svr_names[j] ) {
452                     trouve=1
453                 }
454             }
455             if(trouve == 0) {
456                 nb_of_distinct_svr_name += 1
457                 svr_names[nb_of_distinct_svr_name] = $1
458                 print $1
459             }
460         }
461     }
462 }
463 END {if(sub("^exit ","", line)) exit line; exit -1 }'
464 exit $?
465 ;;
466 svrseq)
467     appname=$1
468     set_enviro
469     get_tuxconfig | \
470     sed -e "s/= /g" -e 's/"//g' -e 's/\\/\\/' -e "s/\\*//" | awk '
471     BEGIN { server=0; nb_of_distinct_svr_seq=0 }
472     {line = $0}
473     $1 == "SEQUENCE" && server == 1 {
474         trouve = 0
475         if (nb_of_distinct_svr_seq == 0) {
476             nb_of_distinct_svr_seq=1
477             svr_seqs[nb_of_distinct_svr_seq] = $2
478             print $2
479         } else {
480             for (j=1; j<= nb_of_distinct_svr_seq; j++) {
481                 if ( $2 == svr_seqs[j] ) {
482                     trouve=1
483                 }
484             }
485             if (trouve == 0) {
486                 nb_of_distinct_svr_seq += 1
487                 svr_seqs[nb_of_distinct_svr_seq] = $2
488                 print $2
489             }
490         }
491     }
492     $1 == "SERVERS" { if($2 == "") server=1}

```

```

493     $1 == "SERVICES" { if($2 == "") server=0)
494     END {if(sub("^exit ", "", line)) exit line; exit -1 }'
495 exit $?
496 ;;
497 svrId)
498     appname=$1
499     set_environ
500     get_tuxconfig | \
501     sed -e "s/= / /g" -e 's// /g' -e 's/\\/ /' -e "s/*//" | awk '
502     BEGIN { server=0; nb_of_distinct_svr_Id=0 }
503     {line = $0}
504     $2 == "SRVGRP" && $4 == "SRVID" && server == 1 {
505         trouve = 0
506         if (nb_of_distinct_svr_Id == 0) {
507             nb_of_distinct_svr_Id=1
508             svr_ids[nb_of_distinct_svr_Id] = $5
509             print $5
510         } else {
511             for (j=1; j<= nb_of_distinct_svr_Id; j++) {
512                 if ( $5 == svr_ids[j] ) {
513                     trouve=1
514                 }
515             }
516             if (trouve == 0) {
517                 nb_of_distinct_svr_Id += 1
518                 svr_ids[nb_of_distinct_svr_Id] = $5
519                 print $5
520             }
521         }
522     }
523     $1 == "SERVERS" { if($2 == "") server=1)
524     $1 == "SERVICES" { if($2 == "") server=0)
525     END {if(sub("^exit ", "", line)) exit line; exit -1 }'
526 exit $?
527 ;;
528 discover_conf)
529     machine=$2
530     appname=$1
531     set_environ
532     get_tuxconfig | \
533     sed -e "s/= / /g" -e 's// /g' -e 's/\\/ /' -e "s/*//" | awk '
534     BEGIN {field = "#"}
535     {line = $0}
536     $1 == "UID" {
537         field = field separator $1
538         value = value separator $2
539         separator = ":"
540     }
541     $1 == "GID" {
542         field = field separator $1
543         value = value separator $2
544         separator = ":"
545     }
546     $1 == "BRIDGE" && network == 1 && mach_found == 1 {
547         field = field separator $1
548         value = value separator $2
549     }
550     $1 == "NLSADDR" && network == 1 && mach_found == 1 {
551         field = field separator $1
552         value = value separator $2
553         network = 0
554         mach_found = 0
555     }
556     $1 == "TLLOG" && $2 == machine {
557         field = field separator $1
558         value = value separator $3
559     }
560     $1 == machine {mach_found = 1}
561     $1 == "NETWORK" { network = 1}
562
563 
```

```

564         END (
565             print field; print value
566             if(sub("^ xit ","", line)) exit line; exit -1
567         )' "machine=$machine"
568     exit $?
569 ;;
570 chglisten)
571     appname=$1
572     machine=$2
573     shift 2
574     if [ $# -gt 0 ]
575     then
576         echo "TLLOG $machine $1" > $ConfDir/tlistenlog.$appname.$machine
577     fi
578     exit $?
579 ;;
580 chklistscript)
581     appname=$1
582     machine=$2
583     set_environ
584     get_tuxval > "appname.tux"
585     get_tllog
586     . ./appname.tux
587     prog="
588 if [ -f $appdir/tlisten.$appname.$machine ]
589 then
590     cat $appdir/tlisten.$appname.$machine
591     echo "\\nexit 0\\"
592 else
593     echo "\\nexit 1\\"
594 fi"
595 if [ -z "$uname" ]
596 then
597     print "Host $machine not found"
598     exit 1
599 fi
600 rm -f tlistscript.$appname.$machine
601 rsh "$uname" -l "$ADMIN" "$prog" | tee tlistscript.$appname.$machine > /
602 dev/null
603 [ $? -ne 0 ] && exit 1
604 [ -s tlistscript.$appname.$machine ] && cat tlistscript.$appname.$machine |
605 awk '
606     END { if ( $2 == "1" ) exit -1 } '
607     [ $? -eq -1 ] && exit 1
608     [ -s tlistscript.$appname.$machine ] && cat tlistscript.$appname.$machine |
609     awk '
610     $1 ~ "tlisten" {
611         mismatch = 0
612         fexec=sprintf("%s/bin/tlisten", tuxdir)
613         if ($1 != fexec) {
614             print "tlisten command full pathnames mismatch"
615             printf "\tscript:\t%s\n", $1
616             printf "\tconfig:\t%s\n", fexec
617             mismatch +=1
618         }
619         for (i=2; i <= NF; i++) {
620             if (( $i == "-d" ) && ( $(i+1) != bridge )) {
621                 print "BRIDGE values mismatch"
622                 printf "\tscript:\t%s\n", $(i+1)
623                 printf "\tconfig:\t%s\n", bridge
624                 mismatch +=1
625             }
626             if (( $i == "-l" ) && ( $(i+1) != nlsaddr )) {
627                 print "NLSADDR values mismatch"
628                 printf "\tscript:\t%s\n", $(i+1)
629                 printf "\tconfig:\t%s\n", nlsaddr
630                 mismatch +=1
631             }
632             if (( $i == "-u" ) && ( $(i+1) != uid )) {
633                 print "UID values mismatch"

```

```

632         printf "\tscript:\t%s\n", $(i+1)
633         printf "\tconfig:\t%s\n",uid
634         mismatch +=1
635     }
636     if (( $i == "-L") && ($(i+1) !=tllog)) {
637         print "LOGFILE values mismatch"
638         printf "\tscript:\t%s\n", $(i+1)
639         printf "\tconfig:\t%s\n", tllog
640         mismatch +=1
641     }
642 }}
643 END {
644     if ( mismatch == 0 )
645         printf "Script File is up-to-date for %s\n",machine
646     else
647         printf "\nScript File is NOT up-to-date for %s\n",machine
648     } ' tllog=$tllog machine=$machine bridge=$bridge \
649         nlsaddr=$nlsaddr uid=$uid tuxdir=$tuxdir
650     exit $?
651 ;;
652 startlistproc)
653     appname=$1; shift
654     list="$*"
655     set_environ
656     boucle_status=0
657     exit_status=0
658     for machine in $list
659     do
660         echo "\n----- Machine: $machine ----- \n"
661         get_tuxval > "appname.tux"
662         get_tllog
663         . ./appname.tux
664         progl="
665         TUXDIR=$tuxdir; export TUXDIR
666         ROOTDIR=$tuxdir; export ROOTDIR # V4
667         APPDIR=$appdir; export APPDIR
668         TUXCONFIG=$tuxconfig; export TUXCONFIG
669         PATH=${PATH}:\$TUXDIR/bin:\$APPDIR; export PATH
670         LANG=$lang; export LANG
671         LIBPATH=${LIBPATH}:\$tuxdir/lib; export LIBPATH
672         COLUMNS=200; export COLUMNS
673         ps -eF '%u %p %a' | awk '\$3 ~ \"tlisten\" && \$0 ~ \"\$nlsaddr\" {
674     exit 1) '
675         if [ \${?} = 1 ]
676         then
677             echo \"Listener already running on $machine\"
678             echo exit 0
679             exit 0
680             fi
681         if [ -f $appdir/tlisten.$appname.$machine ]
682         then
683             . $appdir/tlisten.$appname.$machine
684             ps -eF '%u %p %a' | awk '\$3 ~ \"tlisten\" && \$0 ~ \"\$nls
685     addr\" {exit 1) '
686             if [ \${?} = 1 ]
687             then
688                 echo \"Listener started on $machine\"
689                 echo exit 0
690             else
691                 echo \"Listener starting failed on $machine !!!\"
692                 echo exit 1
693             fi
694         else # create the script file & exec it
695             echo \"\$tuxdir/bin/tlisten -d $bridge -l $nlsaddr -u $uid -L
696             $tllog\" > $appdir/tlisten.$appname.$machine
697             chmod ug+x $appdir/tlisten.$appname.$machine
698             . $appdir/tlisten.$appname.$machine
699             ps -eF '%u %p %a' | awk '\$3 ~ \"tlisten\" && \$0 ~ \"\$nlsadd
700             r\" {exit 1) '
701             if [ \${?} = 1 ]
702             then

```

```

699             echo "\"Listener start d on $machine\"
700             echo  xit 0
701         else
702             echo "\"Listener starting failed on $machine !!!\"
703             echo exit 1
704         fi
705     fi"
706     #echo "$progl" > progl
707     if [ -z "$uname" ]
708     then
709         print "Host $machine not found"
710         exit 1
711     fi
712     rsh "$uname" -l "$ADMIN" "$progl" | awk '
713         NR == 1 {line = $0}
714         NR > 1 { print line; line = $0 }
715         END {if(sub("^exit ","", line)) exit line; print line; exit -1}'
716     boucle_status=`expr $boucle_status \\| $?`
717     done
718     exit $boucle_status
719 ;;
720 stoplistproc)
721     appname=$1; shift
722     list="$*"
723     set_environ
724     boucle_status=0
725     exit_status=0
726     for machine in $list
727     do
728         echo "\n----- Machine: $machine -----\\n"
729         get_tuxval > "appname.tux"
730         . ./appname.tux
731         progl="
732         COLUMNS=200; export COLUMNS
733         ps -eF '%u %p %a' | awk '\\$3 ~ \\\"tlisten\\\" && \\$0 ~ \\\"$nlsaddr\\\" {print \\$
2; exit 0 }' | read pid
734         if [ -n "\\$pid\" ]
735         then
736             kill -9 \\$pid > /dev/null
737             status=\\$?
738             if [ \\$status -eq 0 ]
739             then
740                 echo "\"Process \\$pid killed on $machine\"
741                 echo exit 0
742             else
743                 echo "\"Failed to stop listener on $machine!!!\"
744                 echo exit 1
745             fi
746         else
747             echo "\"No Listener running on $machine\"
748             echo exit 1
749         fi"
750         if [ -z "$uname" ]
751         then
752             print "Host $machine not found"
753             exit 1
754         fi
755         rsh "$uname" -l "$ADMIN" "$progl" | awk '
756             NR == 1 {line = $0}
757             NR > 1 { print line; line = $0 }
758             END {if(sub("^exit ","", line)) exit line; print line; exit -1}'
759         boucle_status=`expr $boucle_status \\| $?`
760         done
761         exit $boucle_status
762     ;;
763
764     runninglist)
765         appname=$1
766         boucle_status=0
767         set_environ
768         list_lmids=`get_tuxconfig | \

```

```

769 sed -e "s=/ /g" -e 's/"//g' -e 's/\\\\/0/' -e "s/\\*//" | awk '
770 BEGIN { network=0 }
771 {line = $0}
772 NF == 1 { if (network == 1) print $1}
773 $1 == "NETWORK" { network = 1}
774 END {if(sub("^exit ", "", line)) exit line; exit -1 }'
775 for machine in $list_lmids
776 do
777     get_tuxval > "appname.tux"
778     . ./appname.tux
779     prog1="
780     TUXDIR=$tuxdir; export TUXDIR
781     LIBPATH=${LIBPATH}:$tuxdir/lib; export LIBPATH
782     ROOTDIR=$tuxdir; export ROOTDIR # V4
783     APPDIR=$appdir; export APPDIR
784     TUXCONFIG=$tuxconfig; export TUXCONFIG
785     PATH=${PATH}:$TUXDIR/bin:$APPDIR; export PATH
786     LANG=$lang; export LANG
787     COLUMNS=200; export COLUMNS
788     ps -ef '%u %p %a' | awk '\$3 ~ \"tlisten\" && \$0 ~ \"\$nlsaddr\" {print
789     \$2}' | read pid
790     if [ -n \"\$pid\" ]
791     then
792         echo \"Listener running on $machine: pid = \$pid\"
793         echo exit 0
794     else
795         echo \"No Listener running on $machine\"
796         echo exit 0
797     fi
798     if [ -z \"$uname\" ]
799     then
800         print \"Host $machine not found\"
801         exit 1
802     fi
803     rsh \"$uname\" -l \"$ADMIN\" \"$prog1\" | awk '
804     NR == 1 {line = $0}
805     NR > 1 { print line; line = $0}
806     END { if (sub(\"^exit \", \"\", line)) exit line; print line; exit -1 }'
807     boucle_status='expr $boucle_status \\| $?'
808 done
809 exit $boucle_status
810 ;;
811 updtlistscript)
812     appname=$1
813     machine=$2
814     set_environ
815     get_tllog
816     get_tuxval > "appname.tux"
817     . ./appname.tux
818     prog="
819     echo \"\$tuxdir/bin/tlisten -d $bridge -l $nlsaddr -u $uid -L $tllog\" > $app
820     dir/tlisten.$appname.$machine
821     chmod ug+x $appdir/tlisten.$appname.$machine
822     echo exit \"\$?\"
823     if [ -z \"$uname\" ]
824     then
825         print \"Host $machine not found\"
826         exit 1
827     fi
828     rsh \"$uname\" -l \"$ADMIN\" \"$prog\" | awk '
829     NR == 1 {line = $0}
830     NR > 1 { print line; line = $0 }
831     END {if(sub(\"^exit \", \"\", line)) exit line; print line; exit -1}'
832     exit $?
833 ;;
834 tuxBootEnt)
835     appname=$1; shift
836     cmd=\"\$TUXDIR/bin/tmboot -y $@\"
837     set_environ
838     remote_cmd
839     exit $?

```



## ANNEXE 2

```

838      ;;
839      tuxShutEnt)
840          appname=$1; shift
841          cmd="\$TUXDIR/bin/tmshutdown -y"
842          set_environ
843          remote_cmd
844          exit $?
845      ;;
846      tuxBootAllMach)
847          appname=$1; shift
848          cmd="\$TUXDIR/bin/tmboot -y -A $@"
849          set_environ
850          remote_cmd
851          exit $?
852      ;;
853      tuxShutAllMach)
854          appname=$1; shift
855          cmd="\$TUXDIR/bin/tmshutdown -y -A $@"
856          set_environ
857          remote_cmd
858          exit $?
859      ;;
860      tuxShut)
861          appname=$1; shift
862          cmd="\$TUXDIR/bin/tmshutdown -y $@"
863          set_environ
864          remote_cmd
865          exit $?
866      ;;
867      tuxShutAdmMast)
868          appname=$1; shift
869          cmd="\$TUXDIR/bin/tmshutdown -y -M $@"
870          set_environ
871          remote_cmd
872          exit $?
873      ;;
874      tuxShutSvrSect)
875          appname=$1; shift
876          cmd="\$TUXDIR/bin/tmshutdown -y -S $@"
877          set_environ
878          remote_cmd
879          exit $?
880      ;;
881      tuxBootAdmMast)
882          appname=$1; shift
883          cmd="\$TUXDIR/bin/tmboot -y -M $@"
884          set_environ
885          remote_cmd
886          exit $?
887      ;;
888      tuxBoot)
889          appname=$1; shift
890          cmd="\$TUXDIR/bin/tmboot -y $@"
891          set_environ
892          remote_cmd
893          exit $?
894      ;;
895      tuxShutdown)
896          appname=$2
897          cmd="\$TUXDIR/bin/tmshutdown -y $1"
898          set_environ
899          remote_cmd
900          exit $?
901      ;;
902      tuxBootSvrSct)
903          appname=$1; shift
904          cmd="\$TUXDIR/bin/tmboot -y -S $@"
905          set_environ
906          remote_cmd
907          exit $?
908      ;;

```

```

909 tuxBootBBL)
910     #echo $*
911     appname=$1; shift
912     cmd="\$TUXDIR/bin/tmboot -y $@"
913     set_environ
914     remote_cmd
915     exit $?
916 ;;
917 tuxShowBooted)
918     appname=$1; shift
919     cmd="(echo psr; echo quit)|\$TUXDIR/bin/tmadmin"
920     set_environ
921     remote_cmd
922     exit $?
923 ;;
924 tuxminIPC)
925     appname=$1; shift
926     cmd="\$TUXDIR/bin/tmboot -y -c $@"
927     set_environ
928     remote_cmd
929     exit $?
930 ;;
931 tuxShutPart)
932     exit_status=0
933     appname=$1;
934     machine=$2; shift
935     set_environ
936     get_tuxconfig | \
937     sed -e "s/=//g" -e 's/"//g' -e 's/\\/\\/' -e "s/\*//" | awk '
938         $1 == "APPDIR" && mach_section == 1 && mach_found == 1 {
939             print "APPDIR " $2 > "appname.tux"
940             mach_section = 0
941             mach_found = 0
942         }
943         $1 == "TUXCONFIG" && mach_section==1 && mach_found==1 {
944             print "TUXCONFIG " $2 > "appname.tux"
945         }
946         $1 == "MACHINES" {mach_section = 1}
947         $2 == "LMID" && mach_section == 1 && $3 == machine {
948             print "MACHINE " $1 > "appname.tux"
949             mach_found = 1
950         }
951         $1 == "TUXDIR" && mach_section==1 && mach_found==1 {
952             print "TUXDIR " $2 > "appname.tux"
953         }
954     ' "machine=$machine" "appname=$appname"
955     if [ $? != 0 ]
956     then
957         exit 1
958     fi
959     appdir=`awk '$1 == "APPDIR" {print $2}' appname.tux`
960     tuxconfig=`awk '$1 == "TUXCONFIG" {print $2}' appname.tux`
961     uname=`awk '$1 == "MACHINE" {print $2}' appname.tux`
962     rootdir=`awk '$1 == "TUXDIR" {print $2}' appname.tux`
963     lang=`sed -e 's/=//g' -e 's/"//g' $ConfDir/$appname.tux |
964         awk '$1 == "LANG" {print $2}'`
965     prog1="TUXDIR=$rootdir; export TUXDIR
966         APPDIR=$appdir; export APPDIR
967         LIBPATH=${LIBPATH}:$rootdir/lib; export LIBPATH
968         TUXCONFIG=$tuxconfig; export TUXCONFIG
969         LANG=$lang; export LANG
970         PATH=${PATH}:$TUXDIR/bin:$APPDIR; export PATH
971         $TUXDIR/bin/tmshutdown -y -P $@
972         echo \$? > /tmp/rem$appname.$machine.tux"
973     if [ -z "$uname" ]
974     then
975         print "Host $machine not found"
976         exit 1
977     fi
978     rsh $uname -l "$ADMIN" "$prog1"
979     rsh_status=`echo $?`

```

## ANNEXE 2

```

980     if [ "$rsh_status" -eq "0" ]
981     then
982         status=`rsh $uname -l "$ADMIN" "cat /tmp/rem$appname.$machine.tux"`
983         rsh $MASTER -l "$ADMIN" "rm /tmp/rem$appname.$machine.tux" 2> /dev/nul
1
984         rsh $uname -l "$ADMIN" "rm /tmp/rem$appname.$machine.tux" 2> /dev/nul
1
985     fi
986     if [ "$status" -ne "0" ]
987     then
988         exit_status=`expr $exit_status + 1`
989     fi
990     if [ "$exit_status" -ne "0" -o "$rsh_status" -ne "0" ]
991     then
992         exit 1
993     fi
994 ;;
995 loadfshm)
996 appname=$1; machine=$2; shift 2
997 set_envIRON
998 get_tuxval > "appname.tux"
999 . ./appname.tux
1000 prog="
1001 TUXDIR=$tuxdir; export TUXDIR
1002 ROOTDIR=$tuxdir; export ROOTDIR
1003 LIBPATH=${LIBPATH}:$tuxdir/lib; export LIBPATH
1004 LANG=$lang; export LANG
1005 $tuxdir/bin/loadfiles $@
1006 echo "\n\nexit \${?}"
1007 if [ -z "$uname" ]
1008 then
1009     print "Host $machine not found"
1010     exit 1
1011 fi
1012 rsh "$uname" -l "$ADMIN" "$prog" | awk '
1013 NR == 1 {line = $0}
1014 NR > 1 { print line; line = $0 }
1015 END {if(sub("^exit ","", line)) exit line; print line; exit -1}'
1016 ;;
1017 Unloadcf)
1018 appname=$1
1019 set_envIRON
1020 cmd="\$TUXDIR/bin/tmunloadcf"
1021 if [ $# -eq 2 ]
1022 then
1023     filename=$2
1024     remote_cmd > "$filename"
1025 else
1026     remote_cmd
1027 fi
1028 exit $?
1029 ;;
1030 *)
1031 echo "Command $1 does not exist"
1032 exit 1
1033 ;;
1034 esac

```

- **MAXSERVICES** qui définit le nombre maximum de services pouvant être rattaché à l'application ;

- **OPTIONS** qui permet de préciser si l'application a lieu sur un réseau local ;

5        - **MODEL** qui permet de préciser si l'application est distribuée ou si elle ne l'est pas.

La section machines contient des informations sur chaque machine (puce, trifide, zig, orage) du réseau. Ces informations sont constituées par  
10 les paramètres suivants :

- **LMID** (Logital Machine ID) qui définit le nom logique de la machine c'est-à-dire le nom utilisé en interne par l'application, à la place du nom réseau;

- **TUXDIR** qui spécifie le chemin d'accès au répertoire d'installation  
15 du logiciel "Tuxedo" ;

- **APPDIR** qui spécifie le chemin d'accès aux serveurs applicatifs, c'est-à-dire le chemin menant aux programmes de l'application (par exemple les programmes concernant l'application "TUXEDO") ;

- **TUXCONFIG** qui spécifie le chemin d'accès absolu au fichier  
20 binaire de configuration TUXCONFIG, celui-ci contenant des informations sur l'application ;

- **ENVFILE** qui spécifie le chemin d'accès au fichier contenant les variables d'environnement pour les serveurs et pour les clients d'une machine donnée;

25        - **ULOGPFX** qui spécifie le chemin d'accès au fichier "ULOG" qui contient des informations sur l'historique de l'application.

La section groupe est la section dans laquelle chaque machine est attribuée à un groupe. Dans l'exemple de l'annexe 1, il existe quatre groupes. Un group est un ensemble de serveurs assurant des services

## ANNEXE 1

Nov 20 1997 16:23:57

ubb.dom1

Page 1

```

1  #
2  #       Tuxedo configuration UBBCONFIG for the model TEST1
3  #
4
5  *RESOURCES
6  IPCKEY                191785
7  MASTER                site1
8  DOMAINID              dom1
9  MAXACCESSERS          50
10 MAXSERVERS            50
11 MAXSERVICES           100
12 OPTIONS               LAN
13 MODEL                 MP
14
15 *MACHINES
16 puce                  LMID=site1
17                       TUXDIR="/usr/tuxedo"
18                       APPDIR="/home/dia/tuxedo"
19                       TUXCONFIG="/home/dia/tuxedo/TUXCONFIG"
20                       ENVFILE="/home/dia/tuxedo/envfile_puce"
21                       ULOGPFX="/home/dia/tuxedo/ULOG"
22
23 trifide               LMID=site2
24                       TUXDIR="/usr/tuxedo"
25                       APPDIR="/home/dia/tmp"
26                       TUXCONFIG="/home/dia/tmp/TUXCONFIG"
27                       ENVFILE="/home/dia/tmp/envfile_trifide"
28                       ULOGPFX="/home/dia/tmp/ULOG"
29
30 zig                   LMID=site3
31                       TUXDIR="/usr/tuxedo"
32                       APPDIR="/home/dia/tuxedo"
33                       TUXCONFIG="/home/dia/tuxedo/TUXCONFIG"
34                       ENVFILE="/home/dia/tuxedo/envfile_zig"
35                       ULOGPFX="/home/dia/tuxedo/ULOG"
36
37 orage                 LMID=site4
38                       TUXDIR="/usr/tuxedo"
39                       APPDIR="/home/dia/tuxedo"
40                       TUXCONFIG="/home/dia/tuxedo/TUXCONFIG"
41                       ENVFILE="/home/dia/tuxedo/envfile_orage"
42                       ULOGPFX="/home/dia/tuxedo/ULOG"
43
44
45
46 *GROUPS
47
48 DEFAULT:              TMSNAME=TMS      TMSCOUNT=2
49 GROUP1                 LMID=site1
50                       GRPNO=1
51 GROUP2                 LMID=site2
52                       GRPNO=2
53 GROUP4                 LMID=site3
54                       GRPNO=3
55 GROUP3                 LMID=site4
56                       GRPNO=4
57
58
59 *SERVERS
60 #
61 DEFAULT: RESTART=Y MAXGEN=5 REPLYQ=Y CLOPT="-A"
62
63 SRV1
64                       SRVGRP=GROUP1
65                       SRVID=100
66                       MIN=2    MAX=2
67                       RQADDR=QSRV1_1
68                       REPLYQ=Y
69                       CLOPT="-s SVC1_1 -s SVC1_2 -- "
70 SRV2
71                       SRVGRP=GROUP2

```

Nov 20 1997 16:23:57

ubb.dom1

Page 2

```

72          SRVID=200
73          MIN=2   MAX=2
74          RQADDR=QSRV2_2
75          REPLYQ=Y
76          CLOPT="--s SVC2_1 -s SVC2_2 -- "
77  SRV4
78          SRVGRP=GROUP4
79          SRVID=300
80          MIN=2   MAX=2
81          RQADDR=QSRV4_3
82          REPLYQ=Y
83          CLOPT="--s SVC4_1 -s SVC4_2 -- "
84  SRV3
85          SRVGRP=GROUP3
86          SRVID=400
87          MIN=2   MAX=2
88          RQADDR=QSRV3_4
89          REPLYQ=Y
90          CLOPT="--s SVC3_1 -- "
91
92
93  *SERVICES
94  DEFAULT:      LOAD=50
95  SVC1_1
96  SVC1_2
97  SVC2_1
98  SVC2_2
99  SVC4_1
100 SVC4_2
101 SVC3_1
102
103
104
105 *NETWORK
106 site1
107 #      port number=60951 (ee17 hexa)
108 #      local address=81b683e0
109 #      NADDR="\x0002ee1781b683e00000000000000000"
110 #      BRIDGE="/dev/xti/tcp"
111 #      port number=60952 (ee18 hexa)
112 #      NLSADDR="\x0002ee1881b683e00000000000000000"
113 #
114 site2
115 #      port number=60951 (ee17 hexa)
116 #      local address=81b68387
117 #      NADDR="\x0002ee1781b683870000000000000000"
118 #      BRIDGE="/dev/xti/tcp"
119 #      port number=60952 (ee18 hexa)
120 #      NLSADDR="\x0002ee1881b683870000000000000000"
121 #
122 site3
123 #      port number=60951 (ee17 hexa)
124 #      local address=81b683e1
125 #      NADDR="\x0002ee1781b683e10000000000000000"
126 #      BRIDGE="/dev/xti/tcp"
127 #      port number=60952 (ee18 hexa)
128 #      NLSADDR="\x0002ee1881b683e10000000000000000"
129 #
130 site4
131 #      port number=60951 (ee17 hexa)
132 #      local address=81b6838b
133 #      NADDR="\x0002ee1781b6838b0000000000000000"
134 #      BRIDGE="/dev/xti/tcp"
135 #      port number=60952 (ee18 hexa)
136 #      NLSADDR="\x0002ee1881b6838b0000000000000000"
137 #
138

```

# ANNEXE 2

```

1  # @BULL_COPYRIGHT@
2  #
3  #
4  # HISTORY
5  # $Log: smtuxadmin.ksh,v $
6  # Revision 1.7  1996/02/12  11:40:49  odeadm
7  #      bci V1Set2C 23.01.96
8  #      [1996/01/23  14:31:07  dia]
9  #
10 # Revision 1.6  1995/12/20  14:26:59  odeadm
11 #      V1 Set2: Still troubles with smtuxadmin.ksh
12 #      [1995/12/11  11:56:55  odeadm]
13 #
14 #      07.12.95 V1Set2 first batch of corrections
15 #      [1995/12/07  17:22:57  odeadm]
16 #
17 #      *** empty log message ***
18 #      [1995/11/30  13:48:30  dia]
19 #
20 #      *** empty log message ***
21 #      [1995/11/30  13:48:30  dia]
22 #
23 # Revision 1.5  1995/10/13  11:52:51  odeadm
24 #      Servers TMS/Partitioned mach.
25 #      [1995/10/09  12:05:57  dia]
26 #
27 # Revision 1.4  1995/09/15  15:15:06  odeadm
28 #      Corrections MRS BUILD 3
29 #      [1995/09/07  15:45:27  dia]
30 #
31 # Revision 1.3  1995/08/24  13:38:03  odeadm
32 #      Build3
33 #      [1995/08/23  09:04:31  odeadm]
34 #
35 # Revision 1.2  1995/07/19  15:18:13  odeadm
36 #      Madison build M0.2
37 #      [1995/07/10  10:01:58  odeadm]
38 #
39 # $EndLog$
40 #! /bin/ksh
41 ConfDir=$WRAPPING_CONFIGURATION
42 Context=smtuxedo.ctx
43 Scanconf=$MADISON_VAR/surveyor/scanconf.tux
44 V5_to_V4='ROOTDIR=$TUXDIR; export ROOTDIR'
45 Set1_to_Set2='[ -z "$ADMIN" ] && export ADMIN="madison"'
46 cmd=$1; shift
47
48 set_environ() {
49     MASTER=""; APPDIR=""; ADMIN=""
50     filename=$ConfDir/$appname.tux
51     Env=`tuxgetenv -k -v APP_PW $filename << !
52     tuxgetenvp
53     !
54     eval "$Env"; unset APP_PW
55     eval "$Set1_to_Set2"
56     if [ -n "$MASTER" -a -n "$APPDIR" ]
57     then
58         Env="$Env"
59     $PW
60     $Set1_to_Set2
61     $V5_to_V4"
62     LD_LIBRARY_PATH=$LIBPATH; export LD_LIBRARY_PATH;
63     cd $APPDIR
64     PATH=$(PATH):..$APPDIR:$TUXDIR/bin; export PATH'
65     return 0
66     fi
67     exit 1
68 }
69
70 remote_cmd() {
71     prog="$Env

```

```

72 $cmd''
73 status=$?
74 sleep 1
75 echo "\nexit $status"
76 '
77 #print -r "$prog" > prog
78   rsh "$MASTER" -l "$ADMIN" "$prog" | awk '
79       NR == 1 {line = $0}
80       NR > 1 { print line; line = $0}
81       END {if(sub("^exit ", "", line)) exit line; exit -1 }'
82 )
83
84
85 get_tuxconfig() {
86     if [ -s tuxconf.tmp.$appname ]
87     then
88         cat tuxconf.tmp.$appname
89     else
90         rm -f tuxconf.tmp.*
91         prog="$Env"
92 $TUXDIR/bin/tmunloadcf
93 echo "\nexit $?"
94 '
95 #print -r "$prog" > prog
96     rsh "$MASTER" -l "$ADMIN" "$prog" | tee tuxconf.tmp.$appname
97     fi
98 get_tlistenlog
99 }
100
101 get_tlistenlog() {
102     tllogfname=$ConfDir/tlistenlog.$appname.$machine
103 if [ -s $tllogfname ]
104 then
105     cat $tllogfname
106 else # default value
107     echo "TLLOG $machine $MADISON_TMP/tlisten.$appname.$machine.log" | tee $tllogfname
108 fi
109     echo "\nexit $?"
110 }
111
112 get_tuxval() {
113     get_tuxconfig | \
114     sed -e "s/= / /g" -e 's/"//g' -e 's/\\\\/0/g' | awk '
115 BEGIN {
116     tuxconfig_section["*RESOURCES"] = 1
117     tuxconfig_section["*MACHINES"] = 2
118     tuxconfig_section["*GROUPS"] = 3
119     tuxconfig_section["*SERVERS"] = 4
120     tuxconfig_section["*SERVICES"] = 5
121     tuxconfig_section["*ROUTING"] = 6
122     tuxconfig_section["*NETWORK"] = 7
123 }
124 NF == 1 {
125     if ( $1 in tuxconfig_section ) {
126         section = tuxconfig_section[$1]
127         next
128     }
129 }
130 section == 2 && $2 == "LMID" { # MACHINES section
131 if ( $3 == machine) {
132     printf "uname=%s\n", $1
133     mach_found=1
134 }
135 else { # reset mach_found for furtheur machines
136     mach_found = 0
137 }
138 next
139 }
140 section == 2 && $1=="APPDIR" && mach_found==1 {
141     printf "appdir=%s\n", $2
142     appdir = $2

```



```

143     next
144   }
145   section == 2 && $1=="TUXCONFIG" && mach_found == 1 {
146     printf "tuxconfig=%s\n", $2
147     next
148   }
149   section == 2 && $1=="TUXDIR" && mach_found==1 {
150     printf "tuxdir=%s\n", $2
151     next
152   }
153   section == 2 && $1=="ROOTDIR" && mach_found==1 { # for V4
154     printf "tuxdir=%s\n", $2
155     next
156   }
157   section == 2 && $1=="ULOGPFX" && mach_found==1 {
158     ulogpfx=1; printf "ulogpfx=%s\n", $2
159     next
160   }
161   section == 7 && NF == 1 {
162     if ( $1 == machine )
163       {mach_found = 1}
164     else { # reset mach_found for other machines
165       mach_found = 0
166     }
167     next
168   }
169   section == 7 && $1=="NLSADDR" && mach_found==1 {
170     printf "nlsaddr=%s\n", $2
171     next
172   }
173   section == 1 && $1 == "UID" {printf "uid=%s\n", $2 ;next }
174   section == 7 && $1=="BRIDGE" && mach_found==1 {
175     printf "bridge=%s\n", $2 }
176   END { # not defined ulogpfx
177     if ( ulogpfx == 0 ) {
178       printf "ulogpfx=%s/ULOG\n", appdir }
179     } ' machine=$machine appname=$appname
180     lang=`sed -e "s=/ /g" -e "s/'/'g" -e "s;/ /" $ConfDir/$appname.tux | awk '
181       $1 == "LANG" {printf "lang=", $2}' `
182   }
183
184   get_tllog() {
185     tllogfname="$ConfDir/tlistenlog.$appname.$machine"
186     if [ -f $tllogfname ]
187     then
188       tllog=`cat $tllogfname|awk '$1 == "TLLOG" && $2 == machine { print $3 }' machine=$machine`
189     else
190       tllog="$MADISON TMP/tlistenlog.$appname.$machine"
191       echo "TLLOG $machine $tllog" > $tllogfname
192     fi
193   }
194
195
196   case $cmd in
197     appli)
198       ls -l $ConfDir 2> /dev/null | awk '
199         sub(".tux$", "", $NF) {print $NF}'
200       ;;
201     isexist)
202       if [ -f $ConfDir/$1.tux ]
203       then
204         echo "Yes"
205       else
206         echo "No"
207       fi
208       ;;
209     setparam)
210       [ ! -d $ConfDir ] && mkdir -p $ConfDir
211       if [ -n "$2" ]
212       then

```

```

213 filename=$ConfDir/$2.tux
214 while [ $# -gt 0 ]
215 do
216     echo "$1=\""$2\""; export $1"
217     shift 2
218 done > $filename
219 fi
220 ;;
221 discover)
222     [ -z "$1" ] && exit 1
223     filename=$ConfDir/$1.tux; shift
224     if [ -f $filename ]
225     then
226         # sed -e 's/://@/g' -e 's/#.*// ' -e 's/ *; */"/g' $filename/ |
227         sed -e 's/#.*// ' -e 's/ *; */"/g' -e 's/://@/g' $filename/
228         | awk ' BEGIN { field = "#promptW:promptP:promptPO:promptS:promptA:pr
229         omptM:promptC:promptR:promptF"; value=":::::::::" }
230         /\=/ {
231             for (i=1; i<= NF; i++) {
232                 if(sub("=$", "", $i)) {
233                     separator = ":"
234                     field = field separator $i
235                     value = value separator $(i+1)
236                 }
237             }
238             END {
239                 print field; print value
240             } FS="'"
241         else
242             print '#\n'
243         fi
244         ;;
245     delapppname)
246         if [ -n "$2" ]
247         then
248             filename=$ConfDir/$2.tux
249             if [ -f $filename ] && grep -q "$1=['\""]*$2" $filename
250             then
251                 rm -f $filename ${filename}p
252             else
253                 echo 'The file does not exist'
254                 echo ' or'
255                 echo 'The file is not an environment file'
256                 exit 1
257             fi
258         fi
259         ;;
260     select)
261         if [ -n "$2" ]
262         then
263             echo "$1='$2'; export $1" > "$Context"
264         fi
265         ;;
266     deselect)
267         rm -f "$Context"
268         ;;
269     selected)
270         APPNAME=""
271         [ -f $Context ] && . ./Context
272         echo "$1$APPNAME"
273         ;;
274     isselected)
275         rm -f tuxconf.tmp.*
276         [ -f $Context ] && fgrep -q "APPNAME=" $Context && shift
277         echo $1
278         ;;
279     loadcf)
280         appname=$1

```

```

281     boucle_status=0
282     cmd="\$TUXDIR/bin/tmloadcf -y $2 $3"
283     set environ
284     echo "---- Loading Configuration Binary Fil ----"
285     remote_cmd
286     status=$?
287     if [ $status -ne 0 ]
288     then
289         exit $status
290     else
291         # maj fichier $Scanconf.tux machines
292         prog="$Env"
293         $TUXDIR/bin/tmunloadcf
294         echo "\nextit $?"
295         '
296         #print -r "$prog" > prog
297         rsh "$MASTER" -l "$ADMIN" "$prog" > tuxconf.tmp.$appname
298         list_lmids='cat tuxconf.tmp.$appname | sed -e "s/= /g" -e "s//g" -e "s/\*//
" | awk '
299         {line = $0}
300         $2 == "LMID" && machine == 1 {lmids = lmids $3 " "; next}
301         $1 == "GROUPS" && $2 == "" { machine=0; next}
302         $1 == "MACHINES" && $2 == "" { machine = 1; next}
303         END {if(sub("^exit ","", line)) {
304             print lmids
305             exit line}
306             exit -1 }'
307         for machine in $list_lmids
308         do
309             echo "---- Updating $Scanconf on $machine ----\n"
310             get_tuxval > "appname.tux"
311             . ./appname.tux
312             log_prefix='echo $ulogpfx | sed -e 's/. .g' | awk '
313             {print $NF} '
314             log_dir='echo $ulogpfx | sed -e 's/. .g' | awk '
315             {for (i=1; i< NF; i++) {
316                 tempo = tempo "/" $i }}
317             END { print tempo}'
318         #Build the 3 lines of $Scanconf for the application
319         prog="
320         [ -x $MADISON_BIN/security/updscantux ] &&
321         $MADISON_BIN/security/updscantux $appname $log_dir $log_prefix
322         echo "\"\nextit \"$?\"\"
323         rsh "$uname" -l madison "$prog" | awk '
324         NR == 1 {line = $0}
325         NR > 1 { print line; line = $0}
326         END {if(sub("^exit ","", line)) exit line; exit -1 }'
327         boucle_status='expr $boucle_status + $?'
328     done
329     fi
330     exit $boucle_status
331     ;;
332     apppwd)
333         filename=$ConfDir/$1.tuxp
334         echo "Enter Application Password: \c"
335         OLDCONFIG='stty -g'
336         stty -echo
337         read APP_PW
338         echo "\nRe-enter Application Password: \c"
339         read APP_PW_1
340         stty $OLDCONFIG
341         if [ "$APP_PW" != "$APP_PW_1" ]
342         then
343             echo "\n\nPassword mismatch!"
344             echo "Enter any character to exit and retry"
345             read
346         else
347             # PWencode "APP_PW=\"\$APP_PW\"; export APP_PW" > $filename
348             # APP_PW='echo $APP_PW | sed -e "s/'/'\"'\"'/g"'
349             # PWencode "APP_PW='\$APP_PW'; xport APP_PW" > $fil name
350     tuxgetenv -s > $filename << !

```

```

351 tuxgetenvp
352 $APP_PW
353 !
354         fi
355         ;;
356     chksyntx)
357         appname=$1
358         cmd="\$TUXDIR/bin/tmloadcf -n $2"
359         set_environ
360         remote_cmd
361         exit $?
362     ;;
363     dispIpc)
364         appname=$1
365         cmd="\$TUXDIR/bin/tmloadcf -c $2"
366         set_environ
367         remote_cmd
368         exit $?
369     ;;
370     machine_network)
371         appname=$1
372         set_environ
373         get_tuxconfig | \
374         sed -e "s/= / /g" -e 's/"//g' -e 's/\\//\' -e "s/\*//\' | awk '
375             BEGIN { network=0 }
376             {line = $0}
377             NF == 1 { if (network == 1) print $1}
378             $1 == "NETWORK" { network = 1}
379             END {if(sub("^exit ", "", line)) exit line; exit -1 }'
380         exit $?
381     ;;
382
383     machine_machines)
384         appname=$1
385         set_environ
386         get_tuxconfig | \
387         sed -e "s/= / /g" -e 's/"//g' -e 's/\\//\' -e "s/\*//\' | awk '
388             BEGIN { machine=0 }
389             {line = $0}
390             $2 == "LMID" { if(machine == 1) print $3}
391             $1 == "GROUPS" { if( $2 == "" ) machine=0}
392             $1 == "MACHINES" { if( $2 == "" ) machine = 1}
393             END {if(sub("^exit ", "", line)) exit line; exit -1 }'
394         exit $?
395     ;;
396     group)
397         appname=$1
398         set_environ
399         get_tuxconfig | \
400         sed -e "s/= / /g" -e 's/"//g' -e 's/\\//\' -e "s/\*//\' | awk '
401             BEGIN { group=0 }
402             {line = $0}
403             $1 == "SERVERS" { group=0 }
404             $1 == "GROUPS" { if($2 == "") group=1}
405             $2 == "LMID" && $4 == "GRPNO" { if(group) print $1}
406             END {if(sub("^exit ", "", line)) exit line; exit -1 }'
407         exit $?
408     ;;
409     svrname)
410         appname=$1
411         set_environ
412         get_tuxconfig | \
413         sed -e "s/= / /g" -e 's/"//g' -e 's/\\//\' -e "s/\*//\' | awk '
414             BEGIN { group=server=nb_of_distinct_svr_name=0 }
415             {line = $0}
416             $1 == "TMSNAME" { if ( group == 1) {
417                 tr uve = 0
418                 if (nb_of_distinct_svr_name == 0) {
419                     nb_of_distinct_svr_name=1
420                     svr_names[nb_of_distinct_svr_name] = $2
421                     print $2

```

```

422     } else {
423         for (j=1; j<= nb_of_distinct_svr_name; j++) {
424             if ( $2 == svr_names[j] ) {
425                 trouve=1
426             }
427         }
428         if (trouve == 0) {
429             nb_of_distinct_svr_name += 1
430             svr_names[nb_of_distinct_svr_name] = $2
431             print $2
432         }
433     }
434 }
435
436 $1 == "SERVERS" { if ($2 == "") {
437     server=1
438     group=0 }
439 }
440 $1 == "SERVICES" { if ($2== "") server=0)
441 $1 == "GROUPS"    { if ($2 == "") group=1)
442 $2 == "SRVGRP" {
443     if((server == 1) && ( $4 == "SRVID")) {
444         trouve = 0
445         if (nb_of_distinct_svr_name == 0) {
446             nb_of_distinct_svr_name = 1
447             svr_names[nb_of_distinct_svr_name] = $1
448             print $1
449         } else {
450             for(j=1; j<= nb_of_distinct_svr_name; j++) {
451                 if ( $1 == svr_names[j] ) {
452                     trouve=1
453                 }
454             }
455             if(trouve == 0) {
456                 nb_of_distinct_svr_name += 1
457                 svr_names[nb_of_distinct_svr_name] = $1
458                 print $1
459             }
460         }
461     }
462 }
463 END {if(sub("^exit .", "", line)) exit line; exit -1 }'
464 exit $?
465 ;;
466 svrseq)
467     appname=$1
468     set_enviro
469     get_tuxconfig | \
470     sed -e "s/=//g" -e 's/"//g' -e 's/\\/\\/' -e "s/\\*/\\/" | awk '
471     BEGIN { server=0; nb_of_distinct_svr_seq=0 }
472     {line = $0}
473     $1 == "SEQUENCE" && server == 1 {
474         trouve = 0
475         if (nb_of_distinct_svr_seq == 0) {
476             nb_of_distinct_svr_seq=1
477             svr_seqs[nb_of_distinct_svr_seq] = $2
478             print $2
479         } else {
480             for (j=1; j<= nb_of_distinct_svr_seq; j++) {
481                 if ( $2 == svr_seqs[j] ) {
482                     trouve=1
483                 }
484             }
485             if (trouve == 0) {
486                 nb_of_distinct_svr_seq += 1
487                 svr_seqs[nb_of_distinct_svr_seq] = $2
488                 print $2
489             }
490         }
491     }
492     $1 == "SERVERS" { if($2 == "") serv r=1)

```

```

493         $1 == "SERVICES" { if($2 == "") server=0}
494     END (if(sub("^exit ", "", line)) xit line; exit -1 )'
495 exit $?
496 ;;
497 svrId)
498     appname=$1
499     set_environ
500     get_tuxconfig | \
501     sed -e "s=/ /g" -e 's"///g' -e 's/\\\\/\\' -e "s/\\*//" | awk '
502         BEGIN { server=0; nb_of_distinct_svr_Id=0 }
503         {line = $0}
504         $2 == "SRVGRP" && $4 == "SRVID" && server == 1 {
505             trouve = 0
506             if (nb_of_distinct_svr_Id == 0) {
507                 nb_of_distinct_svr_Id=1
508                 svr_ids[nb_of_distinct_svr_Id] = $5
509                 print $5
510             } else {
511                 for (j=1; j<= nb_of_distinct_svr_Id; j++) {
512                     if ( $5 == svr_ids[j] ) {
513                         trouve=1
514                     }
515                 }
516                 if (trouve == 0) {
517                     nb_of_distinct_svr_Id += 1
518                     svr_ids[nb_of_distinct_svr_Id] = $5
519                     print $5
520                 }
521             }
522         }
523         $1 == "SERVERS" { if($2 == "") server=1}
524         $1 == "SERVICES" { if($2 == "") server=0}
525     END (if(sub("^exit ", "", line)) exit line; exit -1 )'
526 exit $?
527 ;;
528 discover_conf)
529     machine=$2
530     appname=$1
531     set_environ
532     get_tuxconfig | \
533     sed -e "s=/ /g" -e 's"///g' -e 's/\\\\/\\' -e "s/\\*//" | awk '
534         BEGIN {field = "#"}
535         {line = $0}
536         $1 == "UID" {
537             field = field separator $1
538             value = value separator $2
539             separator = ":"
540         }
541         $1 == "GID" {
542             field = field separator $1
543             value = value separator $2
544             separator = ":"
545         }
546
547         $1 == "BRIDGE" && network == 1 && mach_found == 1 {
548             field = field separator $1
549             value = value separator $2
550         }
551         $1 == "NLSADDR" && network == 1 && mach_found == 1 {
552             field = field separator $1
553             value = value separator $2
554             network = 0
555             mach_found = 0
556         }
557         $1 == "TLLOG" && $2 == machine {
558             field = field separator $1
559             value = value separator $3
560         }
561
562         $1 == machine {mach_found = 1}
563         $1 == "NETWORK" { network = 1}

```

```

564         END (
565             print field; print value
566             if(sub("^exit ", "", line)) exit line; exit -1
567         )' "machine=$machine"
568     exit $?
569     ;;
570 chglisten)
571     appname=$1
572     machine=$2
573     shift 2
574     if [ $# -gt 0 ]
575     then
576         echo "TLLOG $machine $1" > $ConfDir/tlistenlog.$appname.$machine
577     fi
578     exit $?
579     ;;
580 chklistscript)
581     appname=$1
582     machine=$2
583     set_enviro
584     get_tuxval > "appname.tux"
585     get_tllog
586     . ./appname.tux
587     prog="
588     if [ -f $appdir/tlisten.$appname.$machine ]
589     then
590         cat $appdir/tlisten.$appname.$machine
591         echo "\\nexit 0\\"
592     else
593         echo "\\nexit 1\\"
594     fi"
595     if [ -z "$uname" ]
596     then
597         print "Host $machine not found"
598         exit 1
599     fi
600     rm -f tlistscript.$appname.$machine
601     rsh "$uname" -l "$ADMIN" "$prog" | tee tlistscript.$appname.$machine > /
602     dev/null
603     [ $? -ne 0 ] && exit 1
604     [ -s tlistscript.$appname.$machine ] && cat tlistscript.$appname.$machine |
605     awk '
606         END { if ( $2 == "1" ) exit -1 } '
607     [ $? -eq -1 ] && exit 1
608     [ -s tlistscript.$appname.$machine ] && cat tlistscript.$appname.$machine |
609     \
610     awk '
611     $1 ~ "tlisten" {
612         mismatch = 0
613         fexec=sprintf("%s/bin/tlisten", tuxdir)
614         if ($1 != fexec) {
615             print "tlisten command full pathnames mismatch"
616             printf "\tscript:\t%s\n", $1
617             printf "\tconfig:\t%s\n", fexec
618             mismatch +=1
619         }
620         for (i=2; i <= NF; i++) {
621             if (( $i == "-d" ) && ($i+1 != bridge)) {
622                 print "BRIDGE values mismatch"
623                 printf "\tscript:\t%s\n", $i+1
624                 printf "\tconfig:\t%s\n", bridge
625                 mismatch +=1
626             }
627             if (( $i == "-l" ) && ($i+1 != nlsaddr)) {
628                 print "NLSADDR values mismatch"
629                 printf "\tscript:\t%s\n", $i+1
630                 printf "\tconfig:\t%s\n", nlsaddr
631                 mismatch +=1
632             }
633             if (( $i == "-u" ) && ($i+1 != uid)) {
634                 print "UID val u s mismatch"

```

```

632         printf "\tscript:\t%s\n", $(i+1)
633         printf "\tconfig:\t%s\n", uid
634         mismatch +=1
635     }
636     if (( $i == "-L" ) && ($(i+1) !=tllog)) {
637         print "LOGFILE values mismatch"
638         printf "\tscript:\t%s\n", $(i+1)
639         printf "\tconfig:\t%s\n", tllog
640         mismatch +=1
641     }
642     })
643 END {
644     if ( mismatch == 0 )
645         printf "Script File is up-to-date for %s\n", machine
646     else
647         printf "\nScript File is NOT up-to-date for %s\n", machine
648     } ' tllog=$tllog machine=$machine bridge=$bridge \
649         nlsaddr=$nlsaddr uid=$uid tuxdir=$tuxdir
650     exit $?
651 ;;
652 startlistproc)
653     appname=$1; shift
654     list="$*"
655     set_environ
656     boucle_status=0
657     exit_status=0
658     for machine in $list
659     do
660         echo "\n----- Machine: $machine ----- \n"
661         get_tuxval > "appname.tux"
662         get_tllog
663         . ./appname.tux
664         progl="
665         TUXDIR=$tuxdir; export TUXDIR
666         ROOTDIR=$tuxdir; export ROOTDIR # V4
667         APPDIR=$appdir; export APPDIR
668         TUXCONFIG=$tuxconfig; export TUXCONFIG
669         PATH=${PATH}:\$TUXDIR/bin:\$APPDIR; export PATH
670         LANG=$lang; export LANG
671         LIBPATH=${LIBPATH}:\$tuxdir/lib; export LIBPATH
672         COLUMNS=200; export COLUMNS
673         ps -eF '%u %p %a' | awk '\$3 ~ \"tlisten\" && \$0 ~ \"\$nlsaddr\" {
exit 1}'
674         if [ \ $? = 1 ]
675         then
676             echo "\"Listener already running on $machine\"
677             echo exit 0
678             exit 0
679         fi
680         if [ -f $appdir/tlisten.$appname.$machine ]
681         then
682             . $appdir/tlisten.$appname.$machine
683             ps -eF '%u %p %a' | awk '\$3 ~ \"tlisten\" && \$0 ~ \"\$nls
addr\" {exit 1}'
684             if [ \ $? = 1 ]
685             then
686                 echo "\"Listener started on $machine\"
687                 echo exit 0
688             else
689                 echo "\"Listener starting failed on $machine !!!\"
690                 echo exit 1
691             fi
692         else # create the script file & exec it
693             echo "\"$tuxdir/bin/tlisten -d $bridge -l $nlsaddr -u $uid -L
$tllog\" > $appdir/tlisten.$appname.$machine
694             chmod ug+x $appdir/tlisten.$appname.$machine
695             . $appdir/tlisten.$appname.$machine
696             ps -eF '%u %p %a' | awk '\$3 ~ \"tlisten\" && \$0 ~ \"\$nlsadd
r\" {exit 1}'
697             if [ \ $? = 1 ]
698             then

```



```

699             echo "\"Listener started on $machin \""
700             echo exit 0
701         else
702             echo "\"Listener starting failed on $machine !!!\""
703             echo exit 1
704         fi
705     fi"
706     #echo "$progl" > progl
707     if [ -z "$uname" ]
708     then
709         print "Host $machine not found"
710         exit 1
711     fi
712     rsh "$uname" -l "$ADMIN" "$progl" | awk '
713         NR == 1 {line = $0}
714         NR > 1 { print line; line = $0 }
715         END {if(sub("^exit ", "", line)) exit line; print line; exit -1}'
716     boucle_status=`expr $boucle_status \\| $? `
717     done
718     exit $boucle_status
719 ;;
720 stoplistproc)
721     appname=$1; shift
722     list="$*"
723     set_environ
724     boucle_status=0
725     exit_status=0
726     for machine in $list
727     do
728         echo "\n----- Machine: $machine -----\\n"
729         get_tuxval > "appname.tux"
730         . ./appname.tux
731         progl="
732         COLUMNS=200; export COLUMNS
733         ps -ef '%u %p %a' | awk '\\$3 ~ \\\"tlisten\\\" && \\$0 ~ \\\"$nlsaddr\\\" {print \\$
2; exit 0 }' | read pid
734         if [ -n "\\$pid\\\" ]
735         then
736             kill -9 \\$pid > /dev/null
737             status=\\$?
738             if [ \\$status -eq 0 ]
739             then
740                 echo "\"Process \\$pid killed on $machine\""
741                 echo exit 0
742             else
743                 echo "\"Failed to stop listener on $machine!!!\""
744                 echo exit 1
745             fi
746         else
747             echo "\"No Listener running on $machine\""
748             echo exit 1
749         fi"
750         if [ -z "$uname" ]
751         then
752             print "Host $machine not found"
753             exit 1
754         fi
755         rsh "$uname" -l "$ADMIN" "$progl" | awk '
756             NR == 1 {line = $0}
757             NR > 1 { print line; line = $0 }
758             END {if(sub("^exit ", "", line)) exit line; print line; exit -1}'
759         boucle_status=`expr $boucle_status \\| $? `
760         done
761         exit $boucle_status
762     ;;
763     runninglist)
764         appname=$1
765         boucle_status=0
766         set_environ
767         list_lmids=`get_tuxconfig | \

```

```

769 sed -e "s=/ /g" -e 's//g' -e 's/\\\\/0/' -e "s/\\*//" | awk '
770 BEGIN { network=0 }
771 {line = $0}
772 NF == 1 { if (network == 1) print $1}
773 $1 == "NETWORK" { network = 1}
774 END {if(sub("^exit ", "", line)) exit line; exit -1 }'
775 for machine in $list_lmids
776 do
777     get_tuxval > "appname.tux"
778     . ./appname.tux
779     prog1="
780     TUXDIR=$tuxdir; export TUXDIR
781     LIBPATH=${LIBPATH}:$tuxdir/lib; export LIBPATH
782     ROOTDIR=$tuxdir; export ROOTDIR # V4
783     APPDIR=$appdir; export APPDIR
784     TUXCONFIG=$tuxconfig; export TUXCONFIG
785     PATH=${PATH}:$TUXDIR/bin:$APPDIR; export PATH
786     LANG=$lang; export LANG
787     COLUMNS=200; export COLUMNS
788     ps -ef 'tu tp ta' | awk '\$3 ~ \"tlisten\" && \$0 ~ \"\$nlsaddr\" (print
\$2)' | read pid
789     if [ -n \"$pid\" ]
790     then
791         echo \"Listener running on $machine: pid = $pid\"
792         echo exit 0
793     else
794         echo \"No Listener running on $machine\"
795         echo exit 0
796     fi
797     if [ -z \"$uname\" ]
798     then
799         print \"Host $machine not found\"
800         exit 1
801     fi
802     rsh \"$uname\" -l \"$ADMIN\" \"$prog1\" | awk '
803     NR == 1 {line = $0}
804     NR > 1 { print line; line = $0}
805     END { if (sub(\"^exit \", \"\", line)) exit line; print line; exit -1} '
806     boucle_status=`expr $boucle_status \\| $?`
807 done
808 exit $boucle_status
809 ;;
810 updtlistscript)
811 appname=$1
812 machine=$2
813 set_environ
814 get_tllog
815 get_tuxval > "appname.tux"
816 . ./appname.tux
817 prog="
818 echo \"\$tuxdir/bin/tlisten -d $bridge -l $nlsaddr -u $uid -L $tllog\" > $app
dir/tlisten.$appname.$machine
819 chmod ug+x $appdir/tlisten.$appname.$machine
820 echo exit \\$?\"
821 if [ -z \"$uname\" ]
822 then
823     print \"Host $machine not found\"
824     exit 1
825 fi
826 rsh \"$uname\" -l \"$ADMIN\" \"$prog\" | awk '
827 NR == 1 {line = $0}
828 NR > 1 { print line; line = $0 }
829 END {if(sub(\"^exit \", \"\", line)) exit line; print line; exit -1}'
830 exit $?
831 ;;
832 tuxBootEnt)
833 appnam = $1; shift
834 cmd=\"$TUXDIR/bin/tmboot -y $@"
835 set_envir n
836 remote_cmd
837 exit $?

```

```

838      ;;
839      tuxShutEnt)
840          appname=$1; shift
841          cmd="\$TUXDIR/bin/tmshutd wn -y"
842          set_environ
843          remote_cmd
844          exit $?
845      ;;
846      tuxBootAllMach)
847          appname=$1; shift
848          cmd="\$TUXDIR/bin/tmboot -y -A $@"
849          set_environ
850          remote_cmd
851          exit $?
852      ;;
853      tuxShutAllMach)
854          appname=$1; shift
855          cmd="\$TUXDIR/bin/tmshutdown -y -A $@"
856          set_environ
857          remote_cmd
858          exit $?
859      ;;
860      tuxShut)
861          appname=$1; shift
862          cmd="\$TUXDIR/bin/tmshutdown -y $@"
863          set_environ
864          remote_cmd
865          exit $?
866      ;;
867      tuxShutAdmMast)
868          appname=$1; shift
869          cmd="\$TUXDIR/bin/tmshutdown -y -M $@"
870          set_environ
871          remote_cmd
872          exit $?
873      ;;
874      tuxShutSvrSect)
875          appname=$1; shift
876          cmd="\$TUXDIR/bin/tmshutdown -y -S $@"
877          set_environ
878          remote_cmd
879          exit $?
880      ;;
881      tuxBootAdmMast)
882          appname=$1; shift
883          cmd="\$TUXDIR/bin/tmboot -y -M $@"
884          set_environ
885          remote_cmd
886          exit $?
887      ;;
888      tuxBoot)
889          appname=$1; shift
890          cmd="\$TUXDIR/bin/tmboot -y $@"
891          set_environ
892          remote_cmd
893          exit $?
894      ;;
895      tuxShutdown)
896          appname=$2
897          cmd="\$TUXDIR/bin/tmshutdown -y $1"
898          set_environ
899          remote_cmd
900          exit $?
901      ;;
902      tuxBootSvrSct)
903          appname=$1; shift
904          cmd="\$TUXDIR/bin/tmboot -y -S $@"
905          set_environ
906          remote_cmd
907          exit $?
908      ;;

```

```

909 tuxBootBBL)
910     #echo $*
911     appnam=$1; shift
912     cmd="\$TUXDIR/bin/tmboot -y $@"
913     set_environ
914     remote_cmd
915     exit $?
916 ;;
917 tuxShowBooted)
918     appname=$1; shift
919     cmd="(echo psr; echo quit)|\$TUXDIR/bin/tmadmin"
920     set_environ
921     remote_cmd
922     exit $?
923 ;;
924 tuxminIPC)
925     appname=$1; shift
926     cmd="\$TUXDIR/bin/tmboot -y -c $@"
927     set_environ
928     remote_cmd
929     exit $?
930 ;;
931 tuxShutPart)
932     exit_status=0
933     appname=$1;
934     machine=$2; shift
935     set_environ
936     get_tuxconfig | \
937     sed -e "s/= / /g" -e 's/"//g' -e 's/\\// ' -e "s/\\*//" | awk '
938         $1 == "APPDIR" && mach_section == 1 && mach_found == 1 {
939             print "APPDIR " $2 > "appname.tux"
940             mach_section = 0
941             mach_found = 0
942         }
943         $1 == "TUXCONFIG" && mach_section==1 && mach_found==1 {
944             print "TUXCONFIG " $2 > "appname.tux"
945         }
946         $1 == "MACHINES" {mach_section = 1}
947         $2 == "LMID" && mach_section == 1 && $3 == machine {
948             print "MACHINE " $1 > "appname.tux"
949             mach_found = 1
950         }
951         $1 == "TUXDIR" && mach_section==1 && mach_found==1 {
952             print "TUXDIR " $2 > "appname.tux"
953         }
954     ' "machine=$machine" "appname=$appname"
955     if [ $? != 0 ]
956     then
957         exit 1
958     fi
959     appdir=`awk '$1 == "APPDIR" {print $2}' appname.tux`
960     tuxconfig=`awk '$1 == "TUXCONFIG" {print $2}' appname.tux`
961     uname=`awk '$1 == "MACHINE" {print $2}' appname.tux`
962     rootdir=`awk '$1 == "TUXDIR" {print $2}' appname.tux`
963     lang=`sed -e 's/= / /g' -e 's/"//g' -e 's/\\//g' $ConfDir/$appname.tux |
964         awk '$1 == "LANG" {print $2}'`
965     prog1="TUXDIR=$rootdir; export TUXDIR
966         APPDIR=$appdir; export APPDIR
967         LIBPATH=${LIBPATH}:$rootdir/lib; export LIBPATH
968         TUXCONFIG=$tuxconfig; export TUXCONFIG
969         LANG=$lang; export LANG
970         PATH=${PATH}:\$TUXDIR/bin:\$APPDIR; export PATH
971         \$TUXDIR/bin/tmshutdown -y -P $@
972         echo \$? > /tmp/rem$appname.$machine.tux"
973     if [ -z "$uname" ]
974     then
975         print "Host $machine not found"
976         exit 1
977     fi
978     rsh $uname -l "$ADMIN" "$prog1"
979     rsh_status=`echo $?`

```

```

980     if [ "$rsh_status" -eq "0" ]
981     then
982         status=`rsh $uname -l "$ADMIN" "cat /tmp/rem$appname.$machine.tux"`
983         rsh $MASTER -l "$ADMIN" "rm /tmp/rem$appname.$machine.tux" 2> /dev/nul
1
984     rsh $uname -l "$ADMIN" "rm /tmp/rem$appname.$machine.tux" 2> /dev/nul
1
985     fi
986     if [ "$status" -ne "0" ]
987     then
988         exit_status=`expr $exit_status + 1`
989     fi
990     if [ "$exit_status" -ne "0" -o "$rsh_status" -ne "0" ]
991     then
992         exit 1
993     fi
994 ;;
995 loadfshm)
996     appname=$1; machine=$2; shift 2
997     set_environ
998     get_tuxval > "appname.tux"
999     . ./appname.tux
1000     prog="
1001     TUXDIR=$tuxdir; export TUXDIR
1002     ROOTDIR=$tuxdir; export ROOTDIR
1003     LIBPATH=${LIBPATH}:${tuxdir}/lib; export LIBPATH
1004     LANG=$lang; export LANG
1005     $tuxdir/bin/loadfiles $@
1006     echo "\n\nexit \${?}"
1007     if [ -z "$uname" ]
1008     then
1009         print "Host $machine not found"
1010         exit 1
1011     fi
1012     rsh "$uname" -l "$ADMIN" "$prog" | awk '
1013         NR == 1 {line = $0}
1014         NR > 1 { print line; line = $0 }
1015         END {if(sub("^exit ", "", line)) exit line; print line; exit -1}'
1016 ;;
1017 Unloadcf)
1018     appname=$1
1019     set_environ
1020     cmd="\$TUXDIR/bin/tmunloadcf"
1021     if [ $# -eq 2 ]
1022     then
1023         filename=$2
1024         remote_cmd > "$filename"
1025     else
1026         remote_cmd
1027     fi
1028     exit $?
1029 ;;
1030 *)
1031     echo "Command $1 does not exist"
1032     exit 1
1033 ;;
1034 esac

```

## **REVENDEICATIONS**

1. Procédé d'assistance à l'administration d'une application distribuée d'un gestionnaire de traitement des transactions, basée sur un fichier binaire de configuration (TUXCONFIG) caractérisé en ce que ledit procédé comporte:

5       - une étape de décompilation du fichier de configuration actif de la machine maître,

      - une étape de récupération d'informations dans le fichier d configuration décompilé de la machine maître (Mm),

10       - une étape de vérification de la consistance de ladite application mis en oeuvre sur une machine donnée.

2. Procédé d'assistance à l'administration d'une application distribuée d'un gestionnaire de traitement des transactions selon la revendication 1, caractérisé en ce qu'il comprend une étape de gestion d'au moins un module d'écoute (3) d'une machine quelconque de l'application à partir d'une autre machine.

3. Procédé d'assistance à l'administration d'une application distribuée d'un gestionnaire de traitement des transactions selon la revendication 1, caractérisé en ce que les informations concernant ladite application distribuée sont directement prélevées dans le fichier de configuration actif de la machine maître.

4. Procédé d'assistance à l'administration d'une application distribuée d'un gestionnaire de traitement des transactions selon la revendication 1, caractérisé en ce que l'étape de vérification de consistance de ladite application consiste en une comparaison entre des informations issues du fichier de configuration de la machine maître et des informations issues de ladite application courante mise en oeuvre sur une machine donnée.

5. Procédé d'assistance à l'administration d'une application distribuée d'un g stionnaire de traitement d s transactions s lon la revendication 2, caractérisé en ce que la gestion des modules d'écoute consiste à lancer et à arrêter au moins un module d' ' coute, à afficher des informations concernant au

moins un module d'écoute, à modifier le journal d'au moins un module d'écoute, à vérifier le script d'au moins un module d'écoute et à mettre à jour le script d'au moins un module d'écoute.

5 6. Procédé d'assistance à l'administration d'une application distribuée d'un gestionnaire de traitement des transactions selon la revendication 2, caractérisé en ce qu'il comprend une étape de lancement et d'arrêt d'un module d'écoute mis en œuvre sur une première machine, cette étape étant mise en œuvre par un administrateur utilisant une deuxième machine distincte de la première, appartenant au même réseau que la première machine.

10 7. Procédé d'assistance à l'administration d'une application distribué d'un gestionnaire de traitement des transactions selon la revendication 2, caractérisé en ce qu'il comprend une étape d'activation simultanée d plusieurs modules d'écoute.

15 8. Procédé d'assistance à l'administration d'une application d'un moniteur transactionnel basée sur un fichier binaire de configuration selon la revendication 2, caractérisé en ce que les étapes du procédé sont mises en œuvre par l'intermédiaire d'une interface graphique comprenant au moins une icône, au moins un menu, et au moins une boîte de dialogue.

20 9. Procédé d'assistance à l'administration d'une application d'un moniteur transactionnel basée sur un fichier binaire de configuration selon la revendication 8, caractérisé en ce que les menus de l'interface graphique sont structurés sous forme d'arborescence et l'actionnement d'un menu provoque l'affichage d'une liste de valeurs de la configuration courante, sélectionnable par l'utilisateur.

25 10. Procédé d'assistance à l'administration d'une application distribuée d'un gestionnaire de traitement des transactions selon la revendication 4, caractérisé en ce que lorsque le fichier contenant des informations sur ladit application mise en œuvre sur une machine donnée (tlog) est inexistant le procédé le génère automatiquement pour pouvoir l'utiliser lors du prochain  
30 lancement des modules d'écoute (3).

11. Procédure d'assistance à l'administration d'une application distribuée d'un gestionnaire de traitement des transactions selon la revendication 6, caractérisé en ce que lesdites informations affichées concernant au moins un module d'écoute(3) comprennent au moins le nom de ladite application, le nom  
5 logique de la machine (LMID) sur laquelle ladite application est exécutée, l'identification de l'utilisateur (UID) de ladite application, l'adresse utilisée par le module d'écoute (NLSADRR), le chemin d'accès au réseau de ladite application, le chemin d'accès au fichier journal dudit module d'écoute (LLFPN).



obtenu par la sélection du bouton "Command" (25). Le bouton "Reset" (26) permet de réinitialiser les valeurs des pavés (21) et (23). Le bouton "Cancel" (27) permet d'annuler une valeur introduite sur les pavés (21) et (23). Le bouton "?" (28) offre une aide en ligne à l'administrateur.

- 5 Pour chaque machine désignée dans la liste des machines, l'ordinateur se procure des informations sur l'application dans le fichier de configuration de la machine maître et un fichier historique appelé fichier "tlistenlog . <nom de l'application> . <nom de la machine>" contenant des informations sur l'application agissant actuellement sur cette machine.
- 10 L'ordinateur vérifie d'abord si le module d'écoute n'est pas déjà démarré sur la machine. Si c'est le cas, le message "Listener already running on <nom de la machine>" est imprimé sur l'écran. Sinon, si un fichier local existe, l'ordinateur l'exécute et imprime le message "Listener started on the machine", si la commande réussit. Si la commande échoue, l'ordinateur
- 15 imprime le message "Listener starting failed on <nom de la machine>". Si le fichier local n'existe pas, l'ordinateur génère un fichier "tlistenlog . <nom de l'application> . <nom de la machine>" dans le répertoire APPDIR, l'exécute et rend compte du résultat comme précédemment. Ce fichier contient des informations sur l'application courante et sera utilisé dans le prochain
- 20 lancement des modules d'écoute. Ceci correspond aux lignes 652 à 698 de la page 36 et aux lignes 699 à 719 de la page 37 de l'annexe 2.

```

startlistproc)
appname=$1; shift
25     list="$*"
        set_envIRON
        boucle_status=0
        exit_status=0
        for machine in $list
30     do
        echo "\n----- Machine: $machine ----- \n"
        get_tuxval > "appname.tux"
        get_tllog
        . ./appname.tux
35     prog1="
        TUXDIR=$tuxdir; export TUXDIR

```

```

fi
rsh "$uname" -l "$ADMIN" "$prog1" | awk '
    NR == 1 {line = $0}
    NR > 1 { print line; line = $0 }
    END {if(sub("^exit ", "", line)) exit line; print line; exit -1}'
5   boucle_status=`expr $boucle_status \| $? `
done
exit $boucle_status
;;

```

10

Pour arrêter un module d'écoute, l'administrateur sélectionne à partir du menu principal de gestion des modules d'écoute "Manage the Listener Processes", la fonction "Stop Listener Processes" en positionnant son curseur sur la pavé (12) (Figure 1). La fenêtre de la figure 3 apparaît. Elle permet d'indiquer dans un premier pavé (31), le nom de l'application, dans un second pavé (32), le nom de la ou des machines. En cliquant sur le bouton "List" (33), une liste des applications enregistrées ou une liste des machines concernant chaque application peut être obtenue selon la position de la marque de position clignotante (34). Pour chaque machine de l'application, l'ordinateur imprime le nom de la machine pour laquelle le module d'écoute est arrêté. Cette sélection à l'écran grâce à l'interface graphique lance les pas de programmes "stoplistproc" au cours desquels le programme procure à la station sur laquelle la procédure d'arrêt est lancée, des informations par get\_tuxval sur l'application, contenue dans le fichier de configuration de la machine maître (Page 37 de l'Annexe 2, Lignes 720 à 762).

```

stoplistproc)
    appname=$1; shift
30   list="$*"
    set_environ
    boucle_status=0
    exit_status=0
    for machine in $list
35   do
        ech "\n----- Machine: $machine ----- \n"
        get_tuxval > "appname.tux"
        ./appname.tux
    done

```

Id ntifier) appartenant à la configuration du réseau. L'annexe 2 présente aux lignes 764 à 768 de la page 37 et aux lignes 769 à 809 de la page 38, la partie de programme correspondant à l'affichage de la liste des modules d'écoute actifs, qui utilise la fonction get\_tuxval.

5

```

runninglist)
    appname=$1
    boucle_status=0
    set_environ
10    list_lmids=`get_tuxconfig | \
        sed -e "s=/ /g" -e 's/"//g' -e 's/\\V0/' -e "s/^*//" | awk '
            BEGIN { network=0 }
            {line = $0}
            NF == 1 { if (network == 1) print $1}
15    $1 == "NETWORK" { network = 1}
            END {if(sub("^exit ", "", line)) exit line; exit -1 }'`
    for machine in $list_lmids
    do
        get_tuxval > "appname.tux"
20    . ./appname.tux
        prog1="
        TUXDIR=$tuxdir; export TUXDIR
        LIBPATH=${LIBPATH}:$tuxdir/lib; export LIBPATH
        ROOTDIR=$tuxdir; export ROOTDIR # V4
25    APPDIR=$appdir; export APPDIR
        TUXCONFIG=$tuxconfig; export TUXCONFIG
        PATH=${PATH}:$TUXDIR/bin:$APPDIR; export PATH
        LANG=$lang; export LANG
        COLUMNS=200; export COLUMNS
30    ps -eF %u %p %a | awk '$3 ~ /^listen/' && $0 ~ /^$nlsaddr/ {print $2}' |
    read pid
        if [ -n "$pid" ]
        then
            echo "Listener running on $machine: pid = $pid"
35    echo exit 0
        else
            echo "No Listener running on $machine"
            echo exit 0
        fi
40    if [ -z "$uname" ]
        then
            print "Host $machine not found"
            exit 1
        fi
45    rsh "$uname" -l "$ADMIN" "$prog1" | awk '

```

```

        printf "\tconfig:\t%s\n", tlog
        mismatch += 1
    }
}
5  END {
    if ( mismatch == 0 )
        printf "Script File is up-to-date for %s\n", machine
    else
        printf "\nScript File is NOT up-to-date for %s\n", machine
10  } ' tlog=$tlog machine=$machine bridge=$bridge \
    nlsaddr=$nlsaddr uid=$uid tuxdir=$tuxdir
    exit $?
    ;;

```

15        Un script d'un module d'écoute peut aussi être mis à jour par la sélection de la fonction "Update Listener Process scripts to TUXCONFIG Level". Un script d'un module d'écoute Tuxedo permet de lancer un module d'écoute. Il suffit d'intégrer un script de ce type pour une machine donnée, dans la séquence de lancement pour que le module d'écoute soit lancé

20        automatiquement en même temps que la machine. Dans la fenêtre représenté figure 6, l'administrateur entre sur le pavé (61) le nom d'une application, et sur le pavé (62) le nom d'une ou de plusieurs machines. Le programme se procure par l'appel de la sous routine "get\_tuxval", toutes les informations dont il a besoin dans le fichier binaire de configuration extraites

25        par la sous routine "get\_tuxconfig" et crée un fichier lui correspondant dans le répertoire APPDIR sous le nom "tlisten.(nom de l'application).(nom de la machine). Les lignes 810 à 831 de l'annexe 2 page 38 présente la partie du programme correspondant à l'exécution de la commande de mise à jour d'un script d'un module d'écoute.

30

```

updtlistscript)
    appname=$1
    machine=$2
    set_environ
35  get_tlog
    get_tuxval > "appname.tux"
    ./appname.tux
    prog="

```

- le chemin d'accès au pont (BRIDGE) de la machine. Le pont est un processus de gestion des communications entre les serveurs de l'application. Il sert à amorcer l'application. Chaque machine est dotée d'un pont.

5        - l'adresse complète du module d'écoute appelée "NLSADDR". Les quatre premiers chiffres représentent le protocole de communication utilisé. Les quatre chiffres suivants représentent le numéro de port utilisé par le module d'écoute qui doit être différent de celui utilisé par le processus pont (BRIDGE). Les chiffres suivants représentent l'adresse réseau de la machine.

10        La particularité de l'invention est que les informations concernant l'application sont directement prélevées dans le fichier actif de la machine maître. Un administrateur se trouvant sur une machine quelconque du réseau peut gérer l'exécution de la commande "get\_tuxval" sur la machine maître pour le compte de l'administrateur comme représenté en page 27 de l'annexe 2.

15        La sous routine "get\_tuxconfig" du programme utilisé dans la mise en oeuvre du procédé d'assistance à l'administration d'une application distribuée, recherche sur le disque dur de la machine maître le fichier actif de configuration de l'application. Celui-ci est ensuite décompilé au moyen de la commande "tmunloadcf" (Page 28 de Annexe 2, Lignes 85 à 99).

```

get_tuxconfig() {
    if [ -s tuxconf.tmp.$appname ]
25      then
        cat tuxconf.tmp.$appname
    else
        rm -f tuxconf.tmp.*
        prog="$Env"
30  $TUXDIR/bin/tmunloadcf
    echo "\nextit $"
    #print -r "$prog" > prog
        rsh "$MASTER" -l "$ADMIN" "$prog" | tee tuxconf.tmp.$appname
35      fi
    get_tlistenlog

```

}

La sous routine "get\_tuxval" de ce programme (Page 28 de l'annexe 2, lignes 112 à 183) prélève les paramètres tels que LMID, APPDIR, TUXCONFIG, TUXDIR, ROOTDIR, ULOGPFX, NLSADDR, UID et BRIDGE  
 5 du fichier binaire de configuration de l'application obtenue à l'aide de la sous routine "get\_tuxconfig".

```

10 get_tuxval() {
    get_tuxconfig | \
    sed -e "s=/ /g" -e 's"///g' -e 's/\\V0/g' | awk '
  
```

Les valeurs des paramètres recherchées sont tout d'abord initialisées. Pour cela des matrices associatives appelées  
 15 "tuxconfig\_section" sont créées.

```

BEGIN {
    tuxconfig_section["*RESOURCES"] = 1
    tuxconfig_section["*MACHINES"] = 2
    tuxconfig_section["*GROUPS"] = 3
    20 tuxconfig_section["*SERVERS"] = 4
    tuxconfig_section["*SERVICES"] = 5
    tuxconfig_section["*ROUTING"] = 6
    tuxconfig_section["*NETWORK"] = 7
    }
  25
  
```

Un index est associé à chaque matrice. Les paramètres recherchés sont situés dans différentes sections du fichier de configuration. Par exemple pour l'application "Tuxedo", ces différentes sections, au nombre d sept, sont appelées "Ressources", "Machines", "Groupes", "Serveurs",  
 30 "Services" et "Réseau". Pour pouvoir prélever les paramètres dont l'ordinateur a besoin, il doit pouvoir repérer l'endroit où il se trouve dans le fichier de configuration. Dans ce programme, lorsque le nombre de champ (NF) est égal à 1, l'ordinateur se trouve au début d'une section.

```

35 NF == 1 {
    if( $1 in tuxconfig_section ) {
        section = tuxconfig_section[$1]
    }
  
```

Listener Processes", il suffit à l'administrateur de sélectionner la fonction "Change/Show Listener Processes Parameters" sur le pavé (13) de la fenêtre présentée en Figure 1. La fenêtre de la figure 4 apparaît. L'administrateur doit préciser dans le pavé (41), le nom de l'application et dans le pavé (42), un nom de machine. Suite à cette précision, les autres pavés (43 à 46) de la fenêtre font apparaître les valeurs des paramètres tels que :

- l'identification de l'administrateur (UID),
- l'adresse complète du module d'écoute composée de l'adresse de la machine et du numéro de port qu'il utilise (NLSADDR),
- le chemin d'accès au réseau,
- le chemin d'accès complet au fichier journal du module d'écoute (Listener Logfile Full Path Name, LLFPN),

Toutes ces informations sont extraites du fichier TUXCONFIG de la machine maître. Ces informations ne sont pas modifiables par cette commande, à l'exception du LLFPN. L'annexe 2 présente aux lignes 570 à 579 de la page 35, la partie du programme correspondant à l'exécution de la commande de modification du LLFPN.

```

20  chglisten)
      appname=$1
      machine=$2
      shift 2
      if [ $# -gt 0 ]
25      then
          echo "TLLOG $machine $1" > $ConfDir/tlistenlog.$appname.$machine
      fi
      exit $?
      ;;
30

```

Pour pouvoir visualiser les modules d'écoute actifs de l'application, l'administrateur doit sélectionner la fonction "Show currently running Listener Processes" en cliquant sur le pavé (14) de la fenêtre de la Figure 1. L'ordinateur affiche la liste des machines de l'application sur lesquelles un module d'écoute est actif et l'identification du processus PID (Process

```

NR == 1 {line = $0}
NR > 1 { print line; line = $0}
END { if (sub("^exit ", "", line)) exit line; print line; exit -1} '
boucle_status=`expr $boucle_status \|$?`
5  done
  exit $boucle_status
;;

```

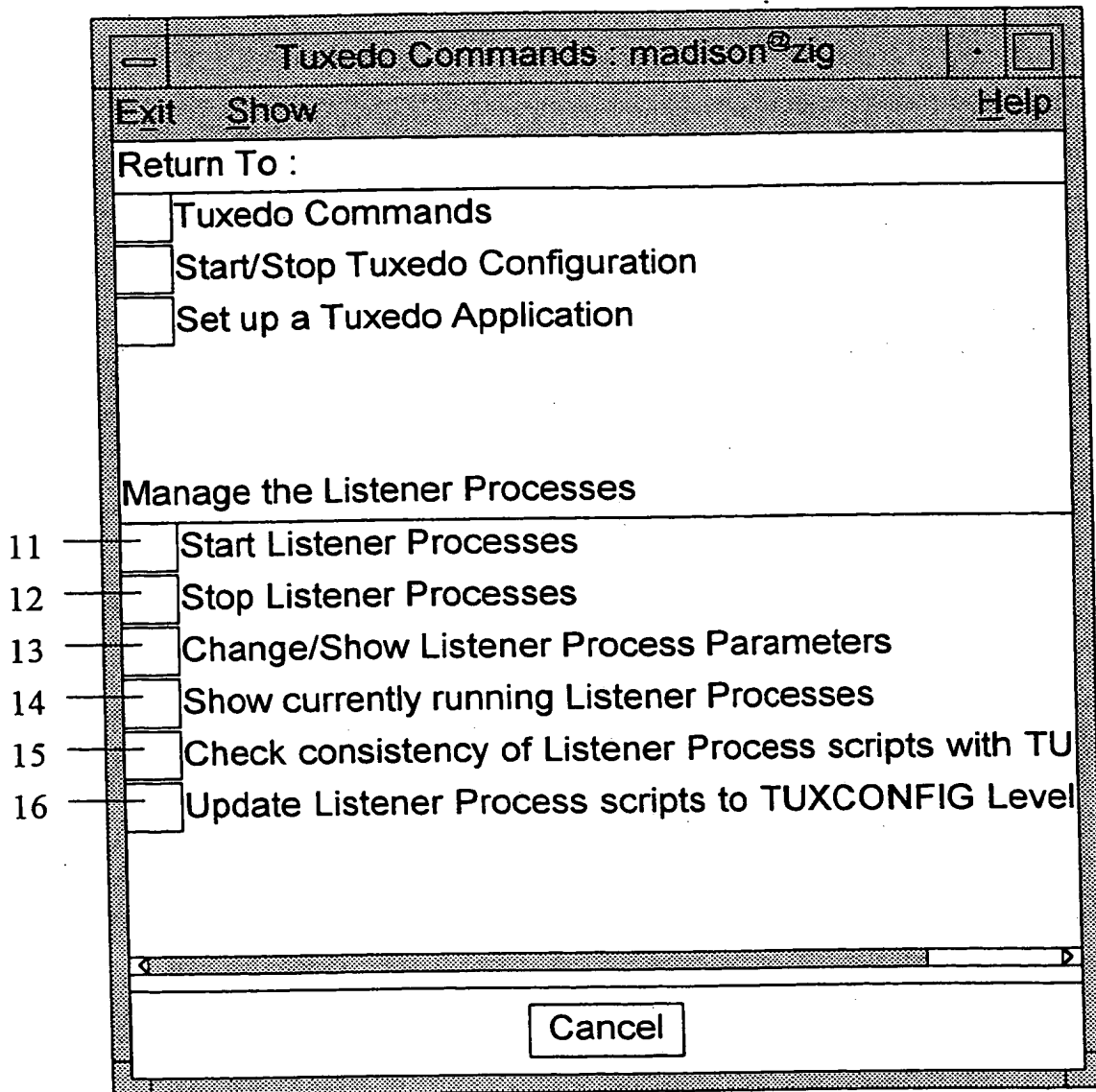
L'administrateur peut aussi vérifier le script d'un module d'écoute. En sélectionnant la fonction "Check consistency of Listener Process scripts with Tuxconfig" sur le pavé (15) de la fenêtre représentée en figure 1, la fenêtre de la figure 5 apparaît. L'administrateur doit entrer le nom d'une application sur le pavé (51) et le nom d'une machine donnée sur le pavé (52). Une liste des applications et des machines est à la disposition de l'administrateur grâce au bouton "List" (53). Le programme compare les informations contenues dans le fichier TUXCONFIG de la machine maître et extraites par la fonction "get\_tuxval" avec les informations contenues dans le fichier "tlisten.(nom de l'application).(nom de la machine)" situé dans le répertoire APPDIR de la machine et donne le résultat de cette comparaison. L'annexe 2 présente aux lignes 580 à 631 de la page 35 et aux lignes 632 à 651 de la page 36, la partie du programme correspondant à la vérification d'un script d'un module d'écoute qui permet de signaler les discordances entre les paramètres des fichiers en imprimant par exemple pour le pont "BRIDGE values mismatch".

```

25  chklistscript)
      appname=$1
      machine=$2
      set_envron
30  get_tuxval > "appname.tux"
      get_tlog
      ./appname.tux
      prog="
      if [ -f $appdir/tlisten.$appname.$machine ]
35  then
          cat $appdir/tlisten.$appname.$machine
          echo "\n\nexit 0"
      else

```



**FIG. 1**

21

22

23

24

25

26

27

28

Start Listener Processes : madison@zig

\* Application Name

\* Machine(s) on which Listener is to be started

dom1

List

OK

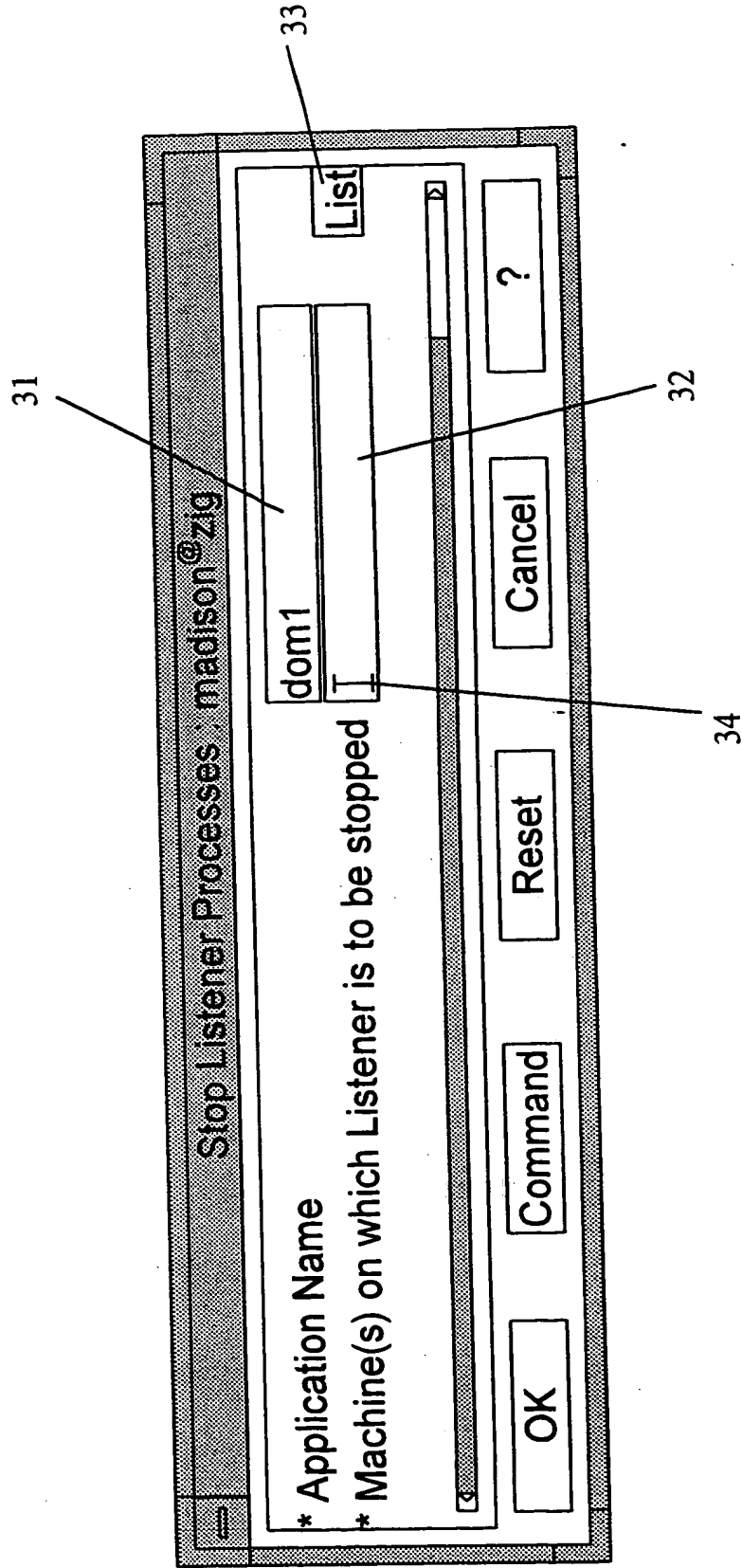
Command

Reset

Cancel

?

**FIG. 2**

**FIG. 3**

Change/Show Listener Processes Parameters madison@zig

* Application Name	dom1	41
* Machine logical name	site1	42
User Identifier	3224	43
Network Listener Address	0x0002ecad81b683e000	44
Network Device Name	/dev/xti/tcp	45
Listener Logfile full path name (Path)	/var/tmp/tlisten.dom	46

OK Command Reset Cancel ?

FIG. 4

Check Listener Processes Script Coherency (Versus Tuxconfig) :

\* Application Name

\* Machine logical name

dom1

List

?

OK

Command

Reset

Cancel

51

53

52

**FIG. 5**

Update Listener Processes Script (Versus Tuxconfig) : madison@zig

\* Application Name

\* Machine logical name

dom1

I

List

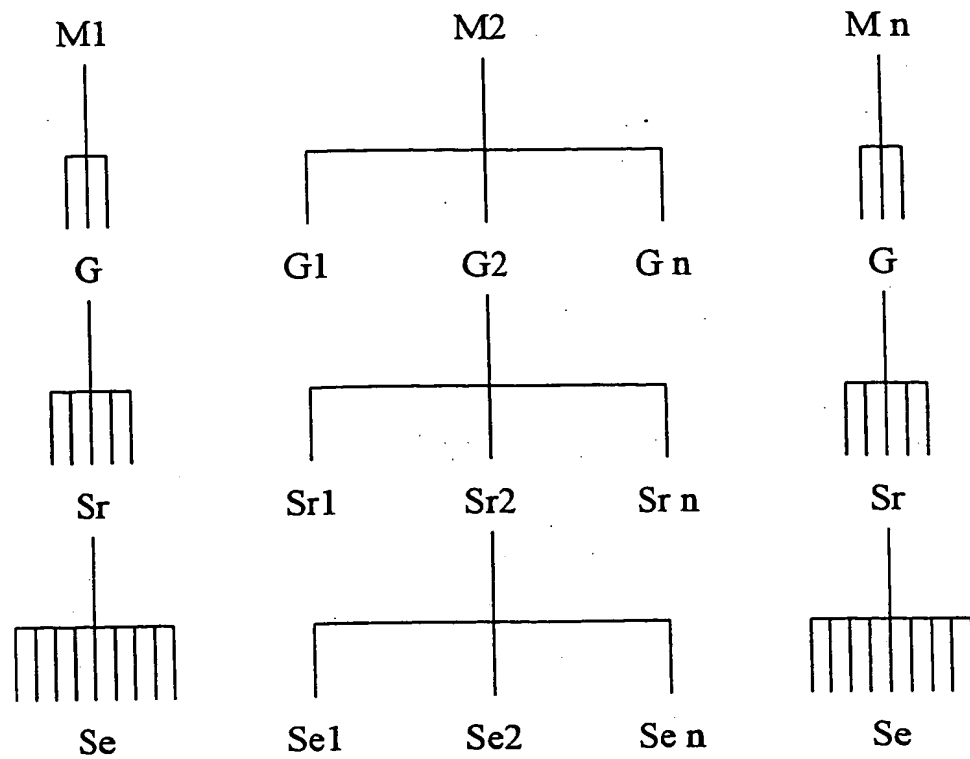
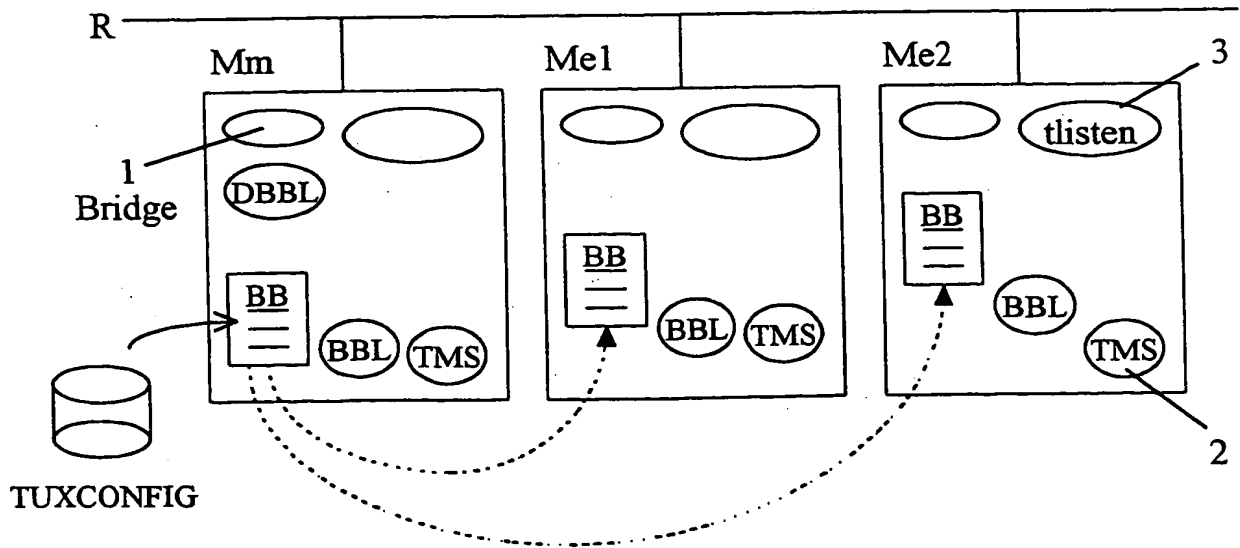
OK Command Reset Cancel ?

61

62

63

**FIG. 6**

**FIG. 7****FIG. 8**

## REVENDEICATIONS

1. Procédé d'assistance à l'administration d'une application distribuée d'un gestionnaire de traitement des transactions, basée sur un  
5 fichier binaire de configuration (TUXCONFIG) caractérisé en ce que ledit procédé comporte:

- une étape de récupération d'informations relatives à ladite application dans un fichier de configuration d'une machine maître (Mm),
  - une étape de vérification de la consistance de ladite application
- 10 mise en oeuvre sur une machine donnée.

2. Procédé selon la revendication 1, caractérisé en ce qu'il comprend une étape de gestion d'au moins un module d'écoute (3) d'une machine quelconque de l'application à partir d'une autre machine.

3. Procédé selon la revendication 1, caractérisé en ce que les  
15 informations concernant ladite application distribuée sont directement prélevées dans le fichier de configuration actif de la machine maître.

4. Procédé selon la revendication 1, caractérisé en ce que l'étape de vérification de consistance de ladite application consiste en une comparaison entre des informations issues du fichier de configuration de la  
20 machine maître et des informations issues de ladite application courante mise en oeuvre sur une machine donnée.

5. Procédé selon la revendication 2, caractérisé en ce que ladite gestion des modules d'écoute consiste à lancer et à arrêter au moins un module d'écoute, à afficher des informations concernant au moins un  
25 module d'écoute, à modifier le journal d'au moins un module d'écoute, à vérifier le script d'au moins un module d'écoute et/ou à mettre à jour le script d'au moins un module d'écoute.

6. Procédé selon la revendication 2, caractérisé en ce qu'il comprend une étape de lancement et d'arrêt d'un module d'écoute mis en  
30 oeuvre sur une première machine, cette étape étant mise en oeuvre par un



administrateur utilisant une deuxième machine distincte de la première, appartenant au même réseau que la première machine.

7. Procédé selon la revendication 2, caractérisé en ce qu'il comprend une étape d'activation simultanée de plusieurs modules d'écoute.
- 5 8. Procédé selon la revendication 1, caractérisé en ce qu'il comporte une étape de décompilation du fichier de configuration actif de la machine maître.
9. Procédé selon la revendication 2, caractérisé en ce que l s étapes du procédé sont mises en œuvre par l'intermédiaire d'une interface
- 10 graphique comprenant au moins une icône, au moins un menu, et au moins une boîte de dialogue.
10. Procédé selon la revendication 9, caractérisé en ce que l s menus de l'interface graphique sont structurés sous forme d'arborescence et l'actionnement d'un menu provoque l'affichage d'une liste de valeurs de la
- 15 configuration courante, sélectionnable par l'utilisateur.
11. Procédé selon la revendication 4, caractérisé en ce que lorsqu le fichier contenant des informations sur ladite application mise en oeuvre sur une machine donnée (tlog) est inexistant le procédé le génère automatiquement pour pouvoir l'utiliser lors du prochain lancement des
- 20 modules d'écoute (3).
12. Procédé selon la revendication 6, caractérisé en ce lesdites informations affichées concernant au moins un module d'écoute(3) comprennent au moins le nom de ladite application, le nom logique de la machine (LMID) sur laquelle ladite application est exécutée, l'identification de
- 25 l'utilisateur (UID) de ladite application, l'adresse utilisée par le module d'écoute (NLSADRR), le chemin d'accès au réseau de ladite application, le chemin d'accès au fichier journal dudit module d'écoute (LLFPN).

**THIS PAGE BLANK (USPTO)**